Exhibit 43

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

EMC CORPORATION,

Petitioner

v.

ACQIS LLC,

Patent Owner

Case No. IPR2014-01462
Patent: 8,041,873

DECLARATION OF VOLKER LINDENSTRUTH

Mail Stop **PATENT BOARD**
Patent Trial and Appeal Board
U.S. Patent and Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450

Table of Contents

I. INTRODUCTION

1. I have been retained by Patent Owner ACQIS LLC to provide this Declaration concerning the technical subject matter relevant to the inter partes review of U.S. Patent No. 8,041,873 ("the '873 patent") (Ex. 1001) entitled "Multiple Module Computer System and Method Including Differential Signal Channel Comprising Unidirectional Serial Bit Channels to Transmit Encoded Peripheral Component Interconnect Bus Transaction Data."

2. The '873 patent claims priority to a provisional application no. 60/134,122 filed on May 14, 1999.

3. I am over 18 years of age. I have personal knowledge of the facts stated in this Declaration and could testify competently to them if asked to do so.

II. PROFESSIONAL QUALIFICATIONS

4. My name is Prof. Dr. Volker Lindenstruth. I am an endowed senior professor for high performance computer architecture at the Goethe-University Frankfurt. I am also a member of the Frankfurt Institute for Advanced Studies (FIAS) which I joined in the year 2007 as a Fellow and later Senior Fellow, and member of the Board of Directors. Since 2012 I have been Chairman of the Board of FIAS. In addition, since 2010 I have headed the Scientific IT at the GSI Helmholtz Center for heavy ion research, an institution with 1,200 employees.

5. I studied physics at the Technical University Darmstadt and completed my PhD in nuclear physics in 1993 at Frankfurt University. I spent the following five years in the US, starting as postdoctoral research fellow at the Lawrence Berkeley National Laboratory and the UC Space Science Laboratory.

6. While I was at Lawrence Berkeley National Laboratory, I collaborated with researchers at the European research center CERN on the RD24 project discussed in the Bogaerts reference. Ex. 1013. I am also a co-author of the Bogaerts reference and I was the principal designer of the CERN/LBL PCI-SCI Adapter discussed in the Bogaerts reference. Ex. 1013.

7. From 1998 to 2009, I held a chair for computer architecture at the University of Heidelberg where I also headed the Kirchhoff Institute for Physics.

8. My research interests concentrate on the fields of computer engineering, high-performance computing and high-energy physics. Since 2000, I have been the chair of the real-time data processing project HLT at the ALICE experiment at the Large Hadron Collider (LHC) accelerator of the European research center CERN.

9. In 2014 my L-CSC supercomputer system, built for the GSI Helmholtz Centre for Heavy Ion Research achieved unprecedented energy efficiency of 5.27 GFlops/W and reached the first place in the world ranking list of the most efficient High Performance Computer systems (the “Green500”). The

L-CSC consists of 160 servers nodes (ASUS ESC4000 G2S/FDR), with 1,600 processor cores; each server contains two Intel-Ivy-Bridge processors and four AMD FirePro graphics cards and has 256 Gigabytes of working memory. The servers are connected through an FDR Infiniband network.

I. MATERIALS CONSIDERED

10. My analysis is based on my experience in the computer industry and educational field since 1993, including documents I have read, research I have conducted, and computer systems I have built during this time.

11. I also reviewed various relevant publications in the computer arts at the time of the invention of the '873 patent. I have also reviewed all the documents provided as exhibits to the '873 patent IPR petition, patent owner's response, the deposition transcript and exhibits of Bruce Young, and the supplementary evidence submitted by the parties. A list of the materials I reviewed in preparing this declaration is set out below:

- Paper No. 2 and Exhibits – EMC's Petition for Inter Partes Review of U.S. Patent No. 8,041,873and attached exhibits.
- Paper No. 11 and Exhibits – ACQIS's Preliminary Patent Owner Response and attached exhibits.
- Paper No. 14 – Board's Institution Decision

- Exhibit 2007 – Horst, TNet: A Reliable System Area Network, 15 IEEE Micro 37 (Feb. 1995) ("Horst") as annotated by Bruce Young.
- Exhibit 2008 – Horst, TNet: A Reliable System Area Network, 15 IEEE Micro 37 (Feb. 1995) ("Horst") as annotated by Bruce Young.
- Exhibit 2009 – Bogaerts, Application of the Scalable Coherent Interface to Data Acquisition at LHC (Oct. 1996) as annotated by Bruce Young.
- Exhibit 2010 – Bogaerts, Application of the Scalable Coherent Interface to Data Acquisition at LHC (Oct. 1996) as annotated by Bruce Young.
- Exhibit 2011 – Declaration of Spencer Scott dated April 8, 2015.
- Exhibit 2012 – Computer Architecture, A Quantitative Approach, J. Hennesy et. al, 1990 (excerpts)
- Exhibit 2013 – Computer Organization and Design , the Hardware/Software Interface, J. Hennesy et al, 1998 (excerpts)
- Exhibit 2014 – PCI Express System Architecture, by R. Budruk, 2004 (excerpts)
- Exhibit 2015 – PCI and PCI-X Hardware and Software Architecture & Design, E. Solari et al, 2001 (excerpts)
- Exhibit 2016 – Intel Nontransparent Bridge Spec., 21554 PCI-to-PCI Bridge

- Exhibit 2017 – VITA Journal, Multiprocessing on PCI and CompactPCI, October/November/December 1998
- Exhibit 2018 – US Patent No. RE42,814
- Exhibit 2019 – PCI Express Revision 3.0, November 2010
- Exhibit 2020 – RapidIO Technology Comparisons - PCIe and Ethernet vs. RapidIO
- Exhibit 2022 – May 20, 2015 Deposition Transcript of B. Young
- Exhibit 2023 – May 21, 2015 Deposition Transcript of B. Young
- Exhibit 2024 – IBM Dictionary of Computing, Tenth Edition, 1993 (excerpts)
- Exhibit 2026 – Declaration of C. Butler from Internet Archives
- In re Lister, 583 F.3d 1307 (Fed. Cir. 2009)
- MPEP sections 2131, 2141, 2143-45
- 2012 Federal Circuit Bar Association Model Patent Jury Instructions
- U.S. Patent No. 8,041,873 File History
- Any documents cited or referenced in this declaration not cited here

II. LEGAL PRINCIPLES USED IN MY ANALYSIS

12. I am not a patent attorney. I have not independently researched the law on patent validity. I have relied on legal principles explained to me by attorneys for the Patent Owner in forming my opinions set forth in this declaration.

B. Validity Analysis Framework—Construe Claims then Compare

13. I understand that analyzing patent validity is a two-part process. The patent claims must first be construed. I discuss the standards for construing the claims below. Second, the claims must be compared to the alleged prior art.

C. Level of Ordinary Skill in the Art

14. I understand that I must base my assessment of the instituted claims of the '873 patent from the perspective of what would have been known or understood by a person having ordinary skill in the art ("POSA") as of the earliest claimed priority date of the patent claim.

15. I understand that, to determine the appropriate level of one of ordinary skill in the art, I may consider the following factors: (a) the types of problems encountered by those working in the field and prior art solutions thereto; (b) the sophistication of the technology in question, and the rapidity with which innovations occur in the field; (c) the educational level of active workers in the field; and (d) the educational level of the inventor.

16. I understand that the relevant technology field for the '873 patent is modular computing systems, computer architecture, and computer communication protocols. Using this field of art and the listed factors above, it is my opinion that a POSA would hold a bachelor's degree or the equivalent in computer science (or related academic fields) and three to four years of additional experience in the field of modular computing systems, computer architecture, or computer communication protocols. This definition of a POSA would not change whether the time of the alleged invention is deemed to be 1998, 1999, or 2000.

17. Unless I state otherwise, when I use "POSA" or "someone of ordinary skill" in my declaration, I am referring to someone with the level of knowledge and understanding set out above.

18. My education and my experience as a professor of computer science and in industry qualify me as a POSA and give me a good understanding of the capabilities of a POSA. I have worked closely with many POSAs over my career. Further, I have regularly taught material fundamental to the art in my role as professor and researcher over the past 20 years.

D. What Constitutes Prior Art

19. I understand that the law sets out categories of information that constitute prior art which may anticipate or render obvious patent claims. To be prior art to a particular patent under the relevant law, I understand a reference must

have been made, known, used, sold, offered for sale, published, or patented, or be the subject of a patent application by another, before the priority date of the patent. I also understand that the POSA is presumed to have knowledge of the relevant prior art.

20. I understand that, for a document to be deemed prior art, it must have been publicly accessible before the priority date of the patent. I understand that publicly accessible requires that a reference must have been sufficiently accessible to the public interested in the art. *In re Lister*, 583 F.3d 1307, 1311 (Fed. Cir. 2009). I understand that a reference is considered publicly accessible if it was (1) publicly disseminated or (2) otherwise made available to the extent that persons interested and ordinarily skilled in the subject matter or art exercising reasonable diligence, could locate it. *In re Lister*, at 1311. I understand that, just because a reference has been stored in a library does not mean the reference is publicly accessible. *In re Lister*, at 1311-12. Instead, I understand that courts have looked to whether the reference was indexed or catalogued by title or subject matter, or were key word searchable. *In re Lister*, at 1312-1317.

E. Claim Construction Standard - Broadest Reasonable Interpretation

21. I understand that, in Inter Partes Review, the claim terms are to be given their broadest reasonable interpretation (BRI) in light of the specification.

See 37 C.F.R. § 42.100(b). In performing my analysis and rendering my opinions, where a claim term has not been construed, I used the ordinary meaning they would have to a the POSA, reading the '873 patent with its priority filing date (May 14, 1999) in mind, and in light of its specification and file history. Where the Board or ACQIS has provided a construction of a term, I applied that construction.

22. Where the Board has construed a term, I have applied the term as construed in my analysis. I understand the Board construed "Peripheral Component Interconnection (PCI) bus transaction" and "PCI bus transaction" to mean the "PCI standard bus transaction." I agree with the Board's construction of PCI bus transaction as relating to a PCI standard bus transaction. As a result, I have based my analysis of "PCI bus transaction" term in the claims on the PCI local bus specification in effect at the time of the earliest priority filing date of the '873 patent. PCI local bus specification version 2.1 was issued in 1995 and remained in effect until late 1998. The PCI local bus specification version 2.2 was issued in late 1998. I have reviewed both the PCI local bus specification version 2.1 and 2.2 and they are materially similar. As a result, I have used the PCI standard set out in the local bus specification revision 2.1 (Ex. 2001) in forming my opinions. I have also relied on my experience with the PCI local bus standard

during the mid-1990s in forming my opinion, including my experience designing the CERN/LBL SCI-PCI Adapter discussed in the Bogaerts reference. Ex. 1013.

23. I understand ACQIS is proposing an additional claim term, "encoded" for construction under BRI. I have set out my analysis of "encoded" and "PCI bus transaction" after the background section below. I believe a discussion of the '873 patent and the background in the art will help inform the Board in evaluating the proper constructions of "encoded" and "a microprocessor coupled to a mass memory storage device."

F. Anticipation

24. I understand that a claim directed to a subject matter that is not new or novel is said to be "anticipated by the prior art" under 35 U.S.C § 102. I understand that a preponderance of the evidence is required to prove that a patent claim is anticipated by a prior art reference in Inter Partes Review.

25. I understand that, in order for a claim to be invalid as anticipated by the prior art, every element of that claim must be found in a single item of prior art. I understand that to be considered prior art it must meet the requirements set out in 35 U.S.C. § 102, and that if a reference does not meet these requirements, it will not be prior art. It is my further understanding that for anticipation, each element of a claim must be found explicitly or inherently in that single item of prior art. As a result, I understand that, in determining whether a single item of prior art

anticipates a patent claim, one should take into account not only what is expressly disclosed in that item, but also what inherently occurred as a natural result of the practice of the system or method disclosed in that item.

G. Obviousness

26. I understand that a patent can be found invalid under 35 U.S.C. § 103 if the subject matter claimed would have been obvious to a person of ordinary skill in the art at the time of the invention. It is also my current understanding that, in assessing the obviousness of the claimed subject matter, one should evaluate obviousness over the prior art from the perspective of one of ordinary skill in the art (and not from the perspective of either a layman or a genius in that art). It is also my current understanding that, in assessing the obviousness of a claimed subject matter, one must evaluate obviousness over the prior art, not in hindsight, but with foresight at the time the invention was conceived. It is my further understanding that the question of obviousness is to be determined based on:

(1) The scope and content of the prior art;

(2) The difference or differences between the subject matter of the claim and the prior art (whereby in assessing the possibility of obviousness one should consider the manner in which a patentee and/or Court has construed the scope of a claim);

(3) The level of ordinary skill in the art at the time of the invention of the subject matter of the claim; and

(4) Any relevant objective factors (the "objective indicia," "secondary indicia," or *Graham Factors*) indicating non-obviousness, including evidence of any of the following:

(a) Commercial success of the products or methods covered by the patent claims;

(b) A long-felt need for the alleged invention;

(c) Failed attempts by others to make the alleged invention;

(d) Copying of the alleged inventions by others in the field;

(e) Unexpected results achieved by the alleged invention;

(f) Praise of the alleged invention by the alleged infringer or other in the field;

(g) The taking of licenses under the patent by others and the nature of those licenses;

(h) The ease with which the patents moved through the PTO;

(i) The presence or lack of some motivation to combine or avoid combining references;

(j) Expression of surprise by experts and those skilled in the art at the subject matter of the claim; and

(k) Whether the patentee proceeded contrary to accepted wisdom of the prior art.

27. I also understand that the United States Supreme Court clarified the law of obviousness in the 2007 *KSR Int'l Co. v. Teleflex Inc.* case ("*KSR*" herein). Based on *KSR*, I understand that, to determine whether it would have been obvious to combine known elements in a manner claimed in a patent, one may consider such things as the interrelated teachings of multiple patents, the effects of demands

known to the design community or present in the marketplace, and the background knowledge of one with ordinary skill in the art.

28. It is further my understanding that, under *KSR*, an obviousness analysis cannot be based solely on conclusory statements, but must include "some articulated reasoning with some rational underpinning to support the legal conclusion of obviousness." I also understand that such reasoning requires at least that an obviousness determination be accompanied by an explanation of the logic that leads to the conclusion that the claimed subject matter is obvious.

III. THE INSTITUTION DECISION

29. I understand that the board instituted Inter Partes Review of claims 54-61of the '873 patent on the following grounds:

Claim	Grounds
54, 57, 60, and 61	Obvious over Horst and the LVDS Owner's Manual (Ground 1)
56 and 59	Obvious over Horst, the LVDS Owner's Manual, and Pocrass (Ground 2)
58	Obvious over Horst, the LVDS Owner's Manual, and Deters (Ground 3)

30. I understand that claim 54 is an independent claim and that claims 56-61 depend from claim 54. As a result, I understand that, if claim 54 is not invalid as anticipated or obvious by the prior art references in the IPR, its dependent claims 56-61 are not anticipated or obvious over those references either.

IV. BACKGROUND

A. Background on the Invention of the '873 Patent

31. The invention claimed in the '873 patent and described in the '873 patent specification relates to ways of interconnecting modular computer systems to computer chassis and I/O devices, motivated by a need to allow central processing units ("CPU"), memory, and storage to be separate from peripheral and input/output ("I/O") devices. Ex. 1001 ('873 patent) at col. 1:30-4:6.

32. An example of this is shown in Figure 1 of the '873 patent.

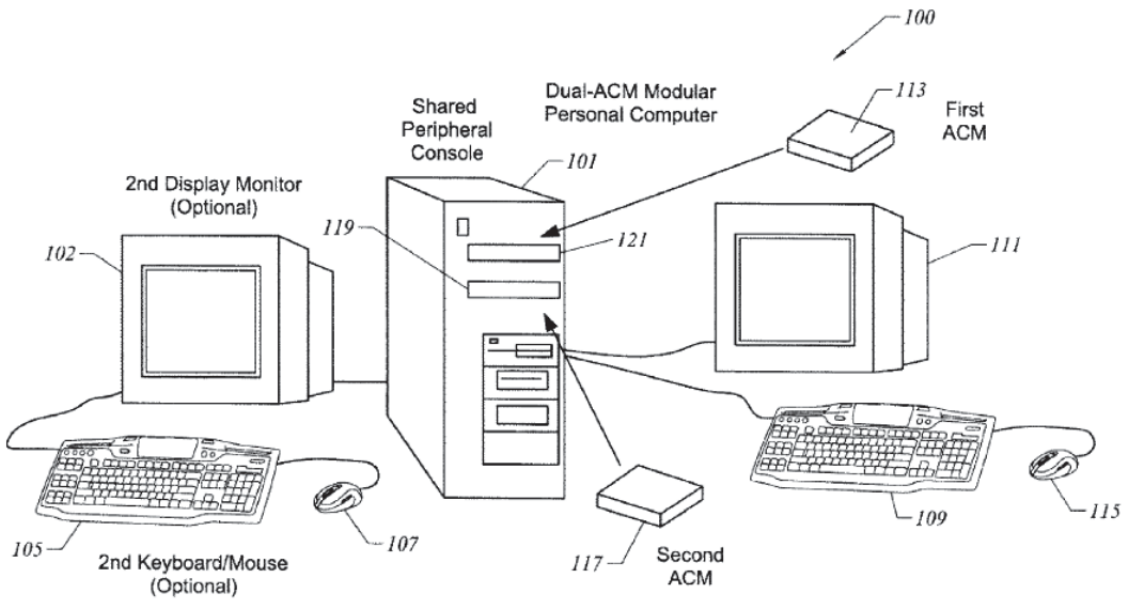


FIGURE 1

33. The computer modules ("ACM") (113, 117) (at least processor and memory) needed to be portable between computer chassis with I/O, power, and peripheral devices (101). Ex. 1001 ('873 patent) at FIG. 1; 1:30-5:24. Figure 20

of the '873 patent shows a block diagram of an example ACM with the connectors (1930a, b) for interconnecting the ACM to the computer console.

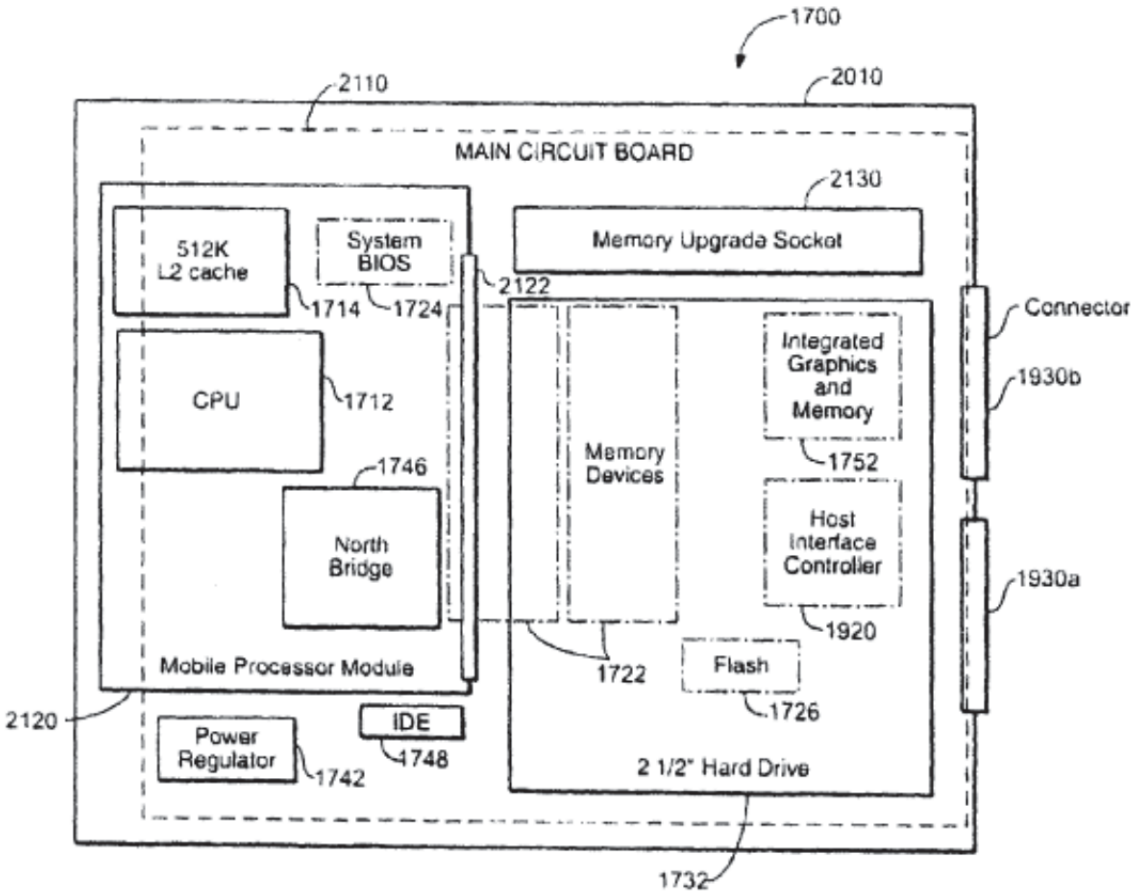
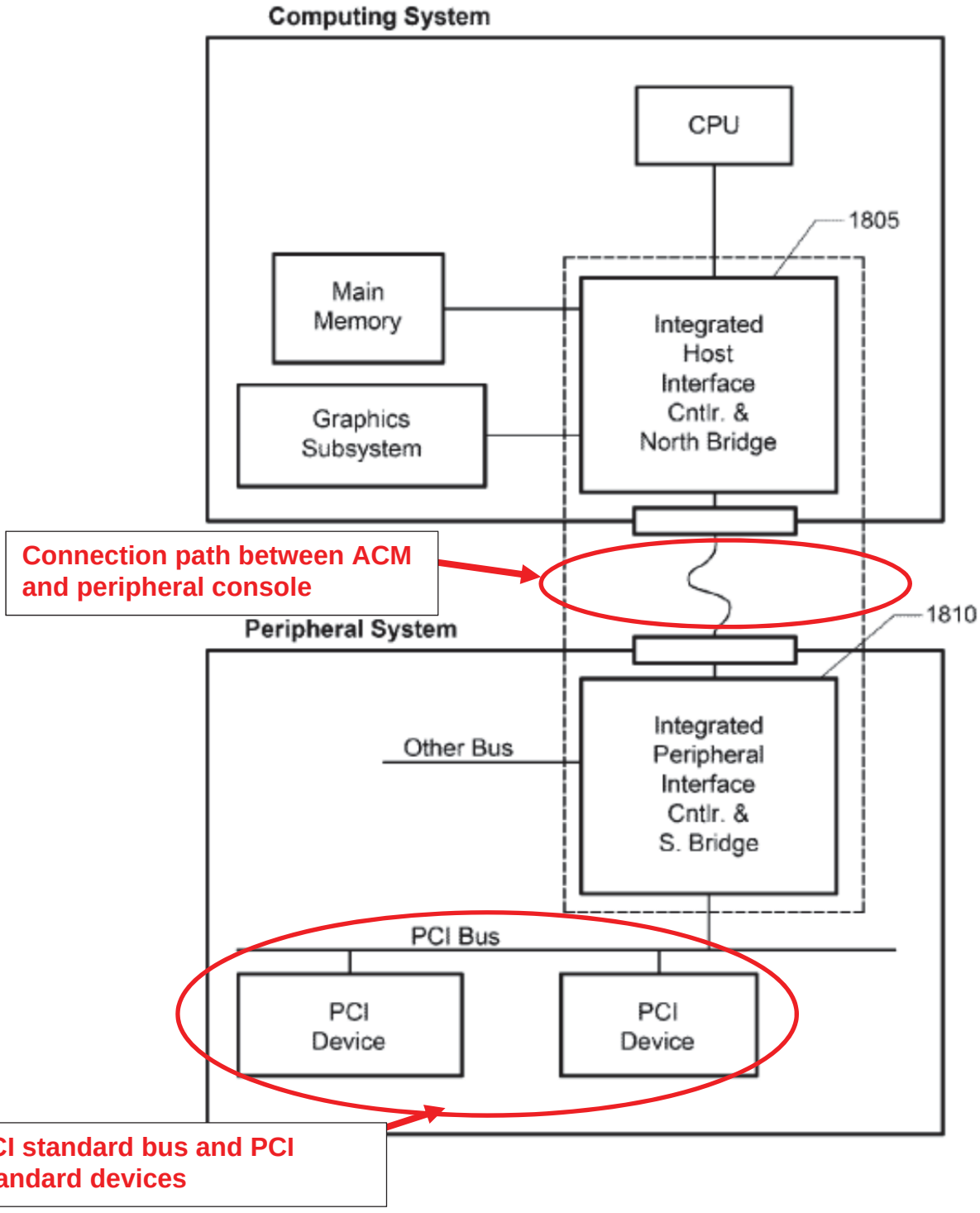


FIGURE 20

34. In order to be acceptable to end users and computer manufacturers, the ACM would need to interact with existing computer components, peripherals, and software drivers without incurring a performance penalty. Ex. 1001 ('873 patent) at FIG. 1; 4:50-67. This required a connection path between the CPU and memory in the ACM and the peripheral components and I/O devices in the console

that was fast and that was cable- and connector-friendly. Ex. 1001 ('873 patent) at FIG. 8, 3:32-67.



35. This required that the ACM be able to communicate over serial lines using an existing parallel bus standard at a speed that could keep up with standard parallel bus speeds. Ex. 1001 ('873 patent) at FIGs. 1, 8; 3:32-67; 4:50-67. As processing speeds were growing exponentially, thanks to Moore's Law, and existing bus speeds were constantly being updated to increase their throughput, to be successful, the ACM's connection had to also allow for speed increases.

B. Computer Architecture

1. Core Components – Processing Unit and Main Memory

36. Computers are devices that can be programmed to execute arithmetic or logical operations automatically. The basic computer has a processing element, generally known as a central processing unit (CPU), memory, and I/O. The CPU executes the arithmetic or logical operations on information stored in memory, and typically stores the results of the operations in memory.

2. Location of Data in a Computer System is Based on an Address

37. Computer processors select the portion of memory to work on through addresses. Addresses define the location of memory, while the information contained in that memory is generally referred to as the data. Depending on the computer system, the memory address space can have a different size. The size of a computer system's address space is reflected in the length of the address required to address the entire address space. Older computer systems used a 16-bit address

space; in other words, there were 2^{16} possible memory addresses. More modern computers may have 32 bit, 64 bit, or larger address spaces.

38. Memory addresses can be physical or virtual. Physical addresses relate to a single specific location in memory. The physical memory location has one ascertainable value.

39. Virtual addresses, on the other hand, are generally process-dependent and do not correlate to any particular physical address. Ex. 2012 at 4-9 (Computer Architecture, A Quantitative Approach ("CA") at 433-438). Instead, virtual addresses may cover multiple different memory levels (memory, disk) and are generally assigned as needed for a particular process. Ex. 2012 at 4-9, 11 (CA at 433-438, 460). This means that more than one virtual address may relate to a single physical memory location or a location on disk. Ex. 2012 at 4-9, 11 (CA at 433-438, 460). Similarly, the same virtual address in different processes may relate to different physical memory addresses. Ex. 2012 at 4-9, 11 (CA at 433-438, 460). Further, each virtual-to-physical address translation is a function of time and, therefore, each time a virtual address is translated into a physical address, a different physical address may be generated. Ex. 2012 at 4-9, 11 (CA at 433-438, 460). Virtual addresses may also have a different size address space than the physical memory address space. Ex. 2013 at 3, Fig. 7.21 (Computer Organization and Design (CO&D) at 582).

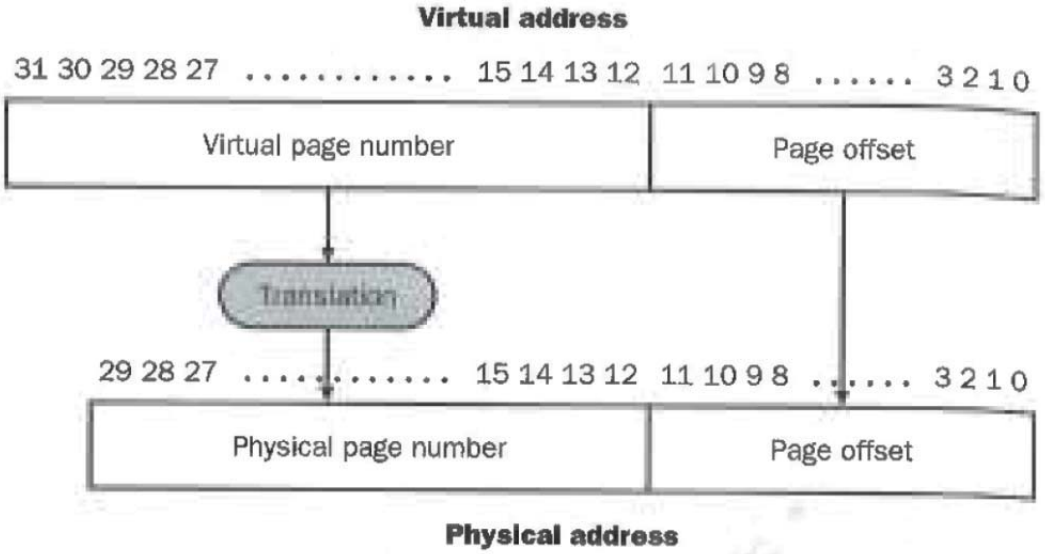


FIGURE 7.21 Mapping from a virtual to a physical address. The page size is 2^{12} = 4 KB. The number of physical pages allowed in memory is 2^{18}, since the physical page number has 18 bits in it. This means that main memory can have at most 1 GB, while the virtual address space is 4 GB.

40. Ex. 2013 at 3 (CO&D at 583). In order to determine which physical address a particular virtual address in a particular process relates to, the virtual address must be mapped by a process page table containing a look-up table to the physical address location. Ex. 2012 at 10 (CA at 433-438, FIG. 8.27). As a result, virtual address mapping is not reversible.

41. In addition, virtual addresses require careful management; otherwise, the processor will read or write the wrong data from or to locations in physical memory.

3. Using the Computer – I/O Devices

42. I/O devices are generally part of the local computer architecture and extend the basic computer system to allow storage and retrieval of information,

display of information, and input and output of information (I/O). Generally, peripheral or I/O devices read and write data to the CPU's main memory and the CPU or the CPU's memory controller will read and write data to the memory of an I/O device. Examples of different types of peripheral devices, their behavior, whether they interact with the user or the processor, and their respective data rates can be found in Figure 8.2 from Computer Organization and Design:

Device	Behavior	Partner	Data rate (KB/sec)
Keyboard	input	human	0.01
Mouse	input	human	0.02
Voice input	input	human	0.02
Scanner	input	human	400.00
Voice output	output	human	0.60
Line printer	output	human	1.00
Laser printer	output	human	200.00
Graphics display	output	human	60,000.00
Modem	input or output	machine	2.00–8.00
Network/LAN	input or output	machine	500.00–6000.00
Floppy disk	storage	machine	100.00
Optical disk	storage	machine	1000.00
Magnetic tape	storage	machine	2000.00
Magnetic disk	storage	machine	2000.00–10,000.00

FIGURE 8.2 The diversity of I/O devices. I/O devices can be distinguished by whether they serve as input, output, or storage devices; their communication partner (people or other computers); and their peak communication rates. The data rates span six orders of magnitude. Note that a network can be an input or an output device, but cannot be used for storage. Disk sizes, as well as transfer rates for devices, are always quoted in base 10, so that 1 MB = 1,000,000 bytes, and 10 Mbit/sec = 10,000,000 bits/sec.

Ex. 2013 at 4 (CO&D at 513).

43. I/O devices are generally much slower than the CPU. As a result, I/O devices, and the connections between the I/O devices and main memory, are often

bottlenecks for CPU and computer-system performance, known as the I/O bottleneck. Ex. 2012 at 14 (CA Fig. 9.11 at 513).

4. Interconnecting the CPU, Memory, and Peripheral and I/O Devices

44. In order for computer systems to function, the various components (CPU, memory, peripherals) must be connected to one another to allow the transfer of information. Figure 9.27 from Computer Organization and Design, below, provides a high level schematic of CPU, main memory, and interconnected I/O peripherals from a computer system in 1997 using a PCI bus (see the arrow).

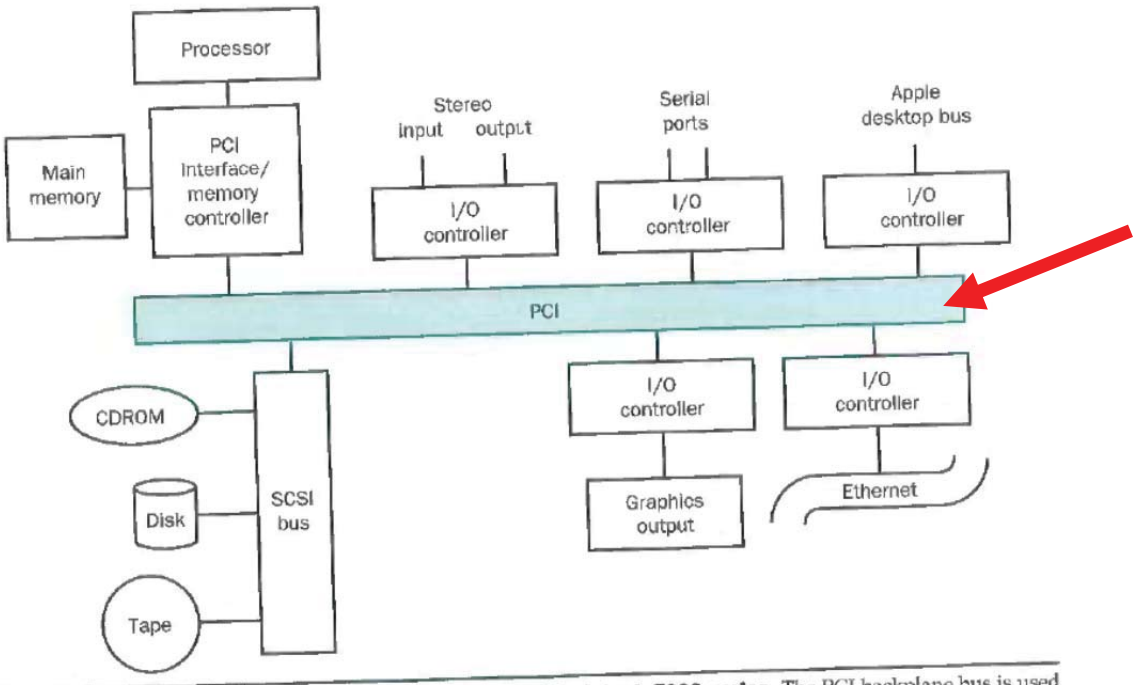


FIGURE 8.16 Organization of the I/O system on the Apple Macintosh 7200 series. The PCI backplane bus is used to interface all devices and interfaces to the processor and memory system. Serial ports provide for connections such as low-speed Appletalk network. The desktop bus provides support for keyboards and mice. In reality, several of the slow I/O devices (audio I/O, serial ports, and the desktop bus) share a single port onto the PCI bus, but we show them separately for simplicity.

Ex. 2013 at 9 (CO&D at 687).

45. The CPU, memory, and its I/O are generally interconnected through busses. At the time of conception of the '873 patent, most computer-system busses were parallel busses. More recently, serial communication point-to-point links, like Intel's QuickPath and PCI Express, have replaced parallel busses. Busses serve as shared communication links between the subsystem of a computer (CPU, memory, I/O devices). As a shared link, multiple devices are connected to the same set of copper traces on a circuit board or channels in an integrated circuit board. The longer a bus, and the more devices connected to it, generally the slower the bus will be to compensate for signal loading and data skew. I will address data skew and communication speed in more detail below.

46. Systems often have more than one type of bus: the CPU-memory bus that interconnects the CPU and memory, and an I/O device bus that connects main memory to I/O devices. Ex. 2012 at18 (CA at 529). The CPU-memory bus, sometimes known as the front-side bus or FSB connects the CPU to memory. The CPU-memory bus is generally very short in length, very high speed and is optimized for the particular processor and memory system. Ex. 2012 at 18 (CA at 529).

47. I/O busses, on the other hand, connect the CPU and main memory (often through a memory controller) to multiple I/O devices. I/O busses typically follow a standard protocol that allows different and new devices to be designed

independently of the CPU and memory system and added later or used with multiple computer systems. Ex. 2012 at 18, 20 (CA at 529, 531). I/O busses are generally slower than CPU-memory busses. As I/O devices are not generally fixed for particular computer systems, bus standards are important as they allow computer systems to be tailored and expanded to meet individual customers' needs. Ex. 2012 at 18, 20 (CA at 529, 531). As long as both the computer-system designer and the I/O-device designer meet the requirements for the bus standard, any I/O device can connect to any computer.

48. When a computer-system grows extremely popular, the I/O bus standard it uses can become a de facto standard through a "snowballing effect" of market dominance. Ex. 2012 at 18, 20 (CA at 529, 531). Many I/O devices will be built for popular computer systems, using that computer system's I/O bus standard. Ex. 2012 at 18, 20 (CA at 529, 531). Once many I/O devices have been built for a computer-system, other computer designers will build their computer systems so that the same I/O devices can be plugged into their computers. Ex. 2012 at 18, 20 (CA at 529, 531). Examples of bus standards that have become de facto standards are the PDP-11 Unibus, the IBM PC-AT Bus, the PCI Local Bus, and PCI Express. Ex. 2012 at 18, 20 (CA at 529, 531); Ex. 2013 at 6 (CD&O at 672).

49. In general, the transfer of information between memory and peripherals on a bus takes the form of a bus transaction, which includes two parts: (i) sending the address and (2) either sending or receiving. Generally, bus transactions are defined by how they act on memory. Ex. 2012 at 18 (CA at 529). Read transactions generally transfer data from the read target; in other words, read transfers read data from a particular memory address or memory address range in main memory or in a peripheral device. Ex. 2012 at 18 (CA at 529). Write transfers, on the other hand, generally transfer data to the write target; in other words, write transfers write data to a particular memory address or memory address range in main memory or in a peripheral device. Ex. 2012 at 18 (CA at 529).

5. Selecting the I/O Device to Communicate With.

50. In order to read or write data to or from an I/O peripheral device, the CPU must be able to address the I/O peripheral and issue one or more of the appropriate command words (e.g., read or write). Ex. 2013 at 8 (CO&D at 675). One way to address I/O devices is called memory-mapped I/O. Ex. 2012 at 22 (CA at 533). Memory-mapped I/O is where portions of the CPU's address space are assigned to I/O devices. Ex. 2012 at 22 (CA at 533). Under memory mapped I/O, to read from or write to a particular I/O device, the CPU simply addresses its memory address space assigned to the I/O device and uses its generic load and

store operations. Ex. 2012 at 22 (CA at 533). The bus controller on the CPU-memory bus interprets this address and command and generates the appropriate bus command on the I/O bus. Ex. 2013 at 8 (CO&D at 675). An example of a typical bus architecture in 1997 involved a front-side bus between the CPU and memory controller and an I/O bus between the memory controller and peripheral devices and other I/O busses. An example is set out in Figure 8.16 from Computer Organization and Design, below:

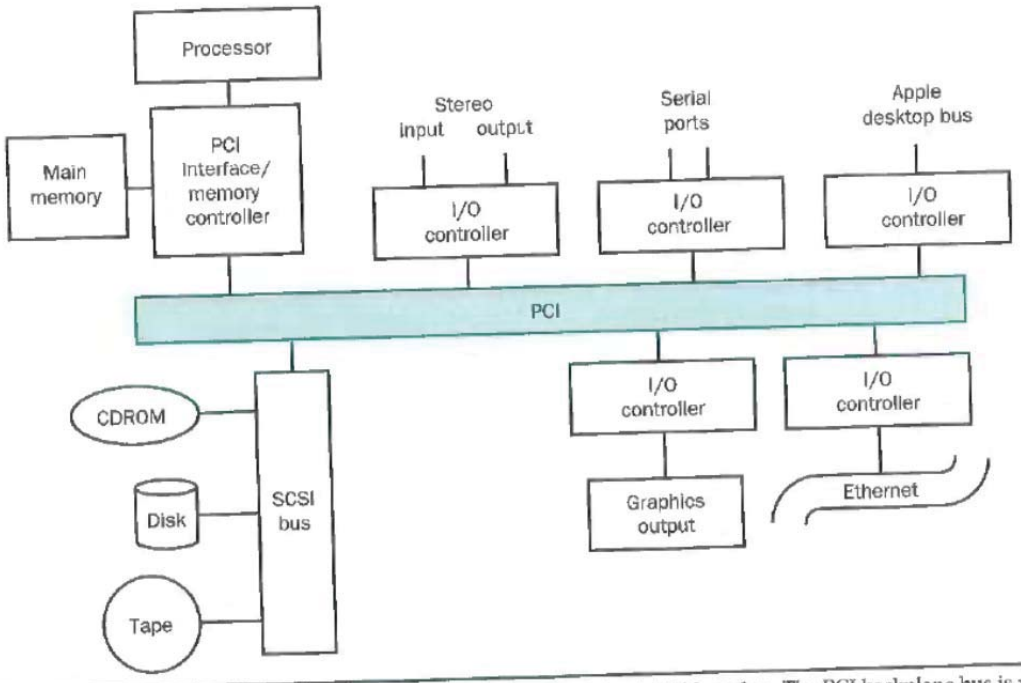


FIGURE 8.16 Organization of the I/O system on the Apple Macintosh 7200 series. The PCI backplane bus is used to interface all devices and interfaces to the processor and memory system. Serial ports provide for connections such as low-speed Appletalk network. The desktop bus provides support for keyboards and mice. In reality, several of the slow I/O devices (audio I/O, serial ports, and the desktop bus) share a single port onto the PCI bus, but we show them separately for simplicity.

Ex. 2013 at 9 (CO&D at 687).

51. Another way to select which I/O device the CPU will communicate with is to use an I/O-specific CPU operation that indicates the address relates to an I/O device. This is often referred to as I/O address space.

6. Networking Devices – Communicating with Other Computers

52. In addition to busses which allow sub-systems within the same computer to communicate, most computers come with network devices connected to the I/O bus that allow the computer to communicate with other computer systems. Ex. 2012 at 15-16 (CA at 526-27). These network devices generally use serial communication protocols which are capable of longer distance connections than parallel communication schemes.

53. An example of a modern network protocol is Ethernet used in Local Area Networks (LANs). Ethernet is a serial protocol with no central master. Ethernet packages data into packets of variable size that are sent across the network. Packets are generally routed using a node or destination identification field in the packet, rather than a memory address.

54. TCP/IP is another network based protocol used widely today in internet communications schemes. TCP/IP breaks up data into packets and operates on a handshake model of data packets and responses.

C. Communication Schemes

1. Parallel Communications

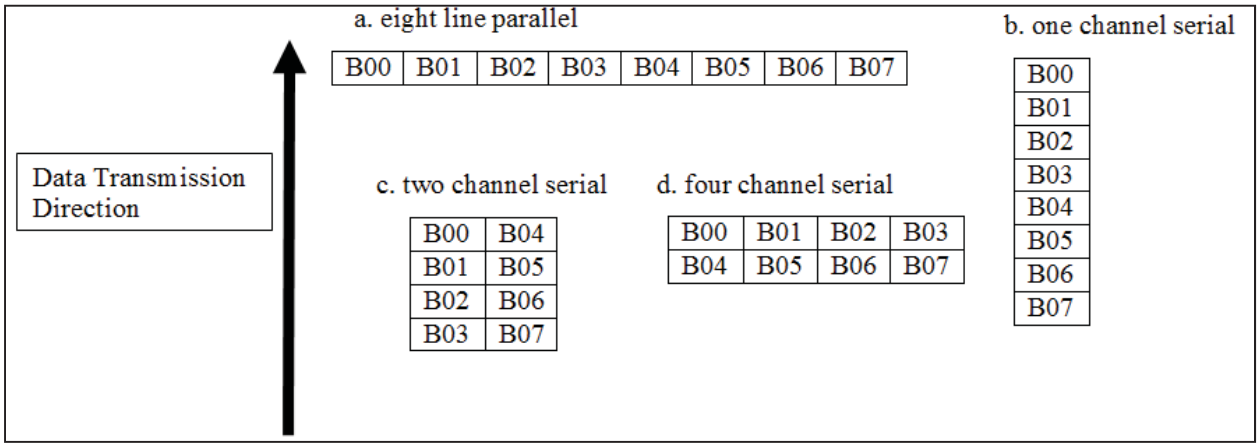
55. Parallel communication protocols involve sending an entire word at once. For example, a 16-bit parallel communication bus would have 16 lines for data and would send 16-bit words every clock cycle data that was being transferred. This is accomplished by driving each of the 16 lines high or low to form an entire 16-bit word. Each line on the parallel bus is said to carry a signal.

56. Parallel busses run into physical limitations on communication speeds as clock speeds increase. Parallel buses are limited in length because, as the length of the bus increases, the difference in length of individual signal lines varies more. Small variations in signal line length can cause signals to arrive at the destination device out of sync. The variation in time it takes signals to arrive at their destination is called skew. Skew requires that parallel busses use a slow enough clock to allow all the signals to arrive at the target device before the target device reads the data to prevent misreading the data, or complex de-skewing hardware is required, which measures and compensates such signal skew.

2. Serial Communications

57. Serial communication protocols send bits of information one after another on one or more lines that are used to make up words. A communication scheme is said to be serial if the information sent over the serial lines has more bits

in a word than there are physical lines to transmit those bits. For example, with a 32-bit word, a communication scheme will be said to be serial if it has less than 32 lines to map the 32 bits onto. The number of lines could be as few as one line, but could commonly be as many as 2, 4, 8, or 16 lines. A diagram of the different possibilities for an 8-bit word sent in parallel, on a single channel serial line, on a 2-channel serial line, and on a 4-channel serial line is set out below for bits B00-B07:



58. The bits in the different configurations can start with either B07 or B00, depending upon convention. Similarly, in the two channel serial (c.) and four channel serial (d.) examples, the bits can be ordered onto the serial channels either vertically (c.) or horizontally (d.).

59. Serial communication schemes do not suffer from data skew as significantly as parallel communication schemes and, therefore, are more suitable for longer interconnects and higher clock speeds. Further, a serial channel will

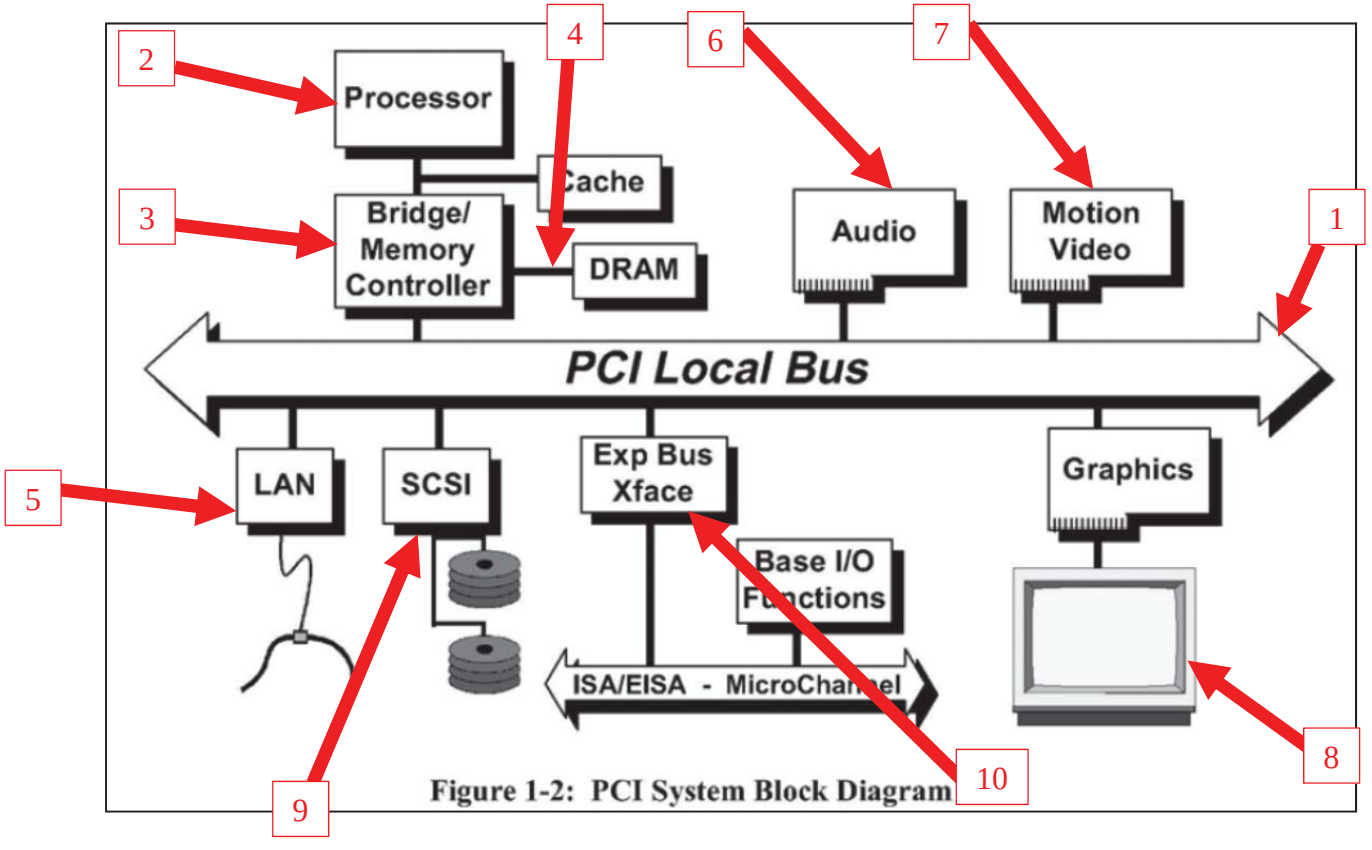
have fewer signal paths for a given word-size than a corresponding parallel bus, meaning that they generally take up less space and use fewer device pins.

3. PCI Local Bus Standard

60. At the time of the conception of the claimed inventions, the peripheral component interconnect (PCI) local bus standard was nearly universal in consumer computers. The PCI standard provided I/O interconnection for processors to their local system and I/O devices. Ex. 2013 at 9 (CO&D at 687). The PCI standard was not designed to network computer systems together. The PCI standard was developed by Intel in the early 1990s and was promulgated and maintained by the PCI-SIG. Ex. 2001 (PCI Spec Rev. 2.1) at 22. At the time the inventions claimed in the '873 patent were conceived, the current PCI standard was at revision 2.1. Ex. 2001.

61. The PCI Local Bus standard was developed to provide a high performance bus for interconnecting highly integrated peripheral components and processor/memory systems. Ex. 2001 (PCI Spec Rev. 2.1) at 17. The motivation for PCI was the development of graphics-oriented systems, such as Windows and OS/2, which created a data bottleneck between the processor and its display peripherals. Ex. 2001 (PCI Spec Rev. 2.1) at 17. The PCI specification teaches that moving peripheral functions with high bandwidth requirements closer to the system's processor bus can help reduce the I/O bottleneck. Ex. 2001 (PCI Spec

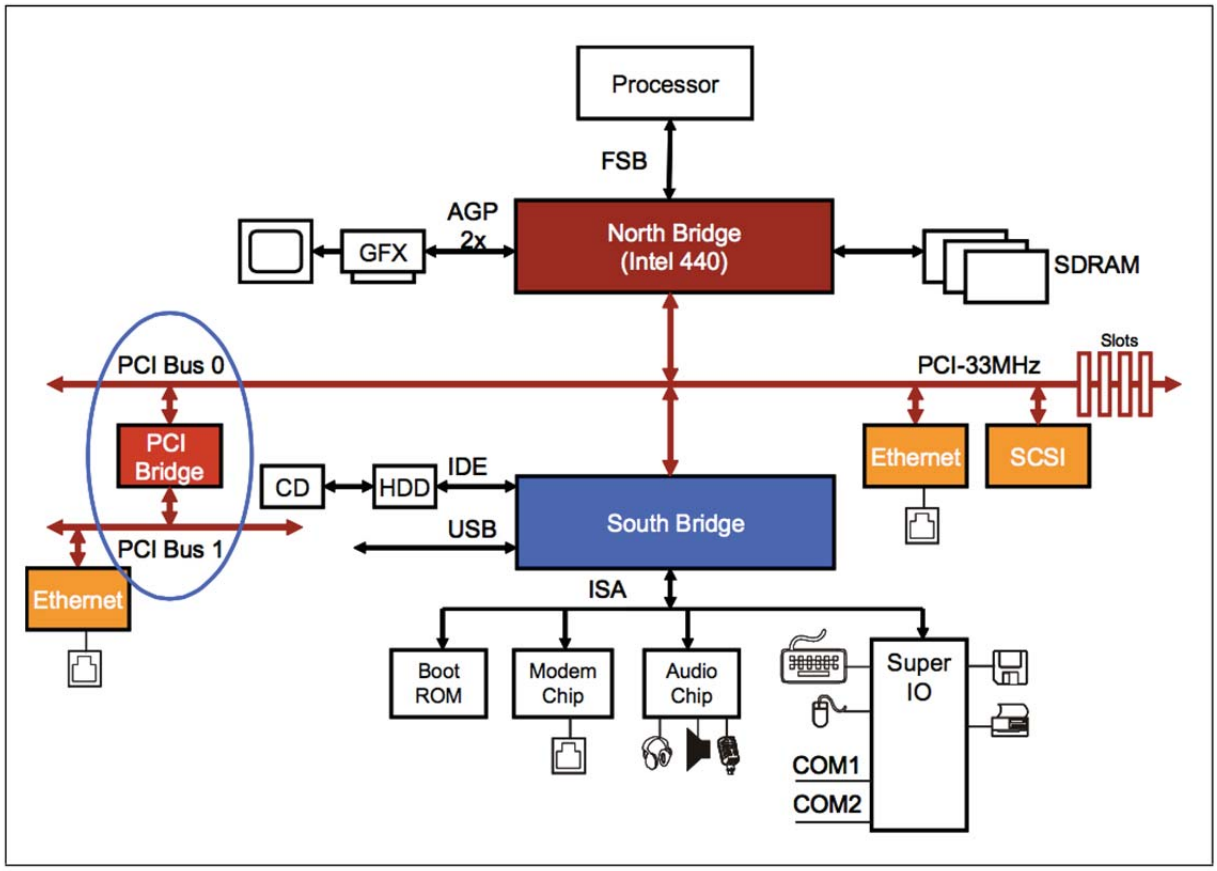
Rev. 2.1) at 17. In other words, the PCI standard is a "local bus." Ex. 2001 (PCI Spec Rev. 2.1) at 17. The PCI standard was focused on processor independence, "enabling an efficient transition to future processor generations and use with multiple processor architectures." Ex. 2001 (PCI Spec Rev. 2.1) at 18. "Processor independence allows the PCI Local Bus to be optimized for I/O functions, enables concurrent operation of the local bus with the processor/memory subsystem, and accommodates multiple high-performance peripherals in addition to graphics (motion video, LAN, SCSI, FDDI, hard disk drives, etc.)." Ex. 2001 (PCI Spec Rev. 2.1) at 18. A typical PCI local bus system architecture is set out below:



Ex. 2001 (PCI Spec Rev. 2.1) at 19.

62. Notice that the PCI Local Bus (1) connects a CPU (the processor)(2) to the I/O peripherals on the computer-system through a Bridge/Memory Controller (3). The Bridge/Memory Controller (3), also known as a North Bridge, generates PCI bus transactions (discussed below) on the PCI bus (1), including PCI bus command, address and data based on the processor addressing an address range corresponding to a device connected to the PCI bus (5, 6, 7, 8, 9, 10) and issuing a command (read or write) on the memory bus (4). Notice also that interconnection between separate computer systems would be handled by the LAN controller (5). Notice, finally, that the PCI bus has components connected directly to the bus (Audio (6), Motion Video (7), Graphics (8)) and connects to different bus controllers that connect to different busses (SCSI (9), Exp Bus Xface (10)). The PCI bus and PCI bus transactions terminate at the LAN, SCSI, and Exp Bus Xface (also known as a "South Bridge") and the respective bus and LAN controllers generate transactions in the respective new formats. The PCI bus does not transmit LAN, SCSI, or Exp Bus Xface commands or addresses; rather, the individual bus or network controller generates the appropriate commands and addresses corresponding to the appropriate protocol based on the PCI bus transaction. Another figure of a typical PCI bus architecture showing a PCI-to-PCI bridge is set out below.

Figure 1-4: 33 MHz PCI Based System Showing Implementation of a PCI-to-PCI Bridge



Ex. 2014 at 5 (PCI Express Sys. Architecture at 19). The PCI to PCI bridge is circled in blue.

63. The PCI standard bus was a 32-bit wide parallel bus with multiplexed address and data bits and separate command and timing control lines. Ex. 2001 (PCI Spec Rev. 2.1) at 17; (a 64-bit wide extension was also part of the PCI specification). The address, data, bus command, and byte enables are described in the portion of the PCI local bus specification set out below:

2.2.2. Address and Data Pins

AD[31::00]	t/s	*Address* and *Data* are multiplexed on the same PCI pins. A bus transaction consists of an address[3] phase followed by one or more data phases. PCI supports both read and write bursts. The address phase is the clock cycle in which **FRAME#** is asserted. During the address phase **AD[31::00]** contain a physical address (32 bits). For I/O, this is a byte address; for configuration and memory, it is a DWORD address. During data phases **AD[07::00]** contain the least significant byte (lsb) and **AD[31::24]** contain the most significant byte (msb). Write data is stable and valid when **IRDY#** is asserted and read data is stable and valid when **TRDY#** is asserted. Data is transferred during those clocks where both **IRDY#** and **TRDY#** are asserted.
C/BE[3::0]#	t/s	*Bus Command* and *Byte Enables* are multiplexed on the same PCI pins. During the address phase of a transaction, **C/BE[3::0]#** define the bus command (refer to Section 3.1. for bus command definitions). During the data phase **C/BE[3::0]#** are used as Byte Enables. The Byte Enables are valid for the entire data phase and determine which byte lanes carry meaningful data. **C/BE[0]#** applies to byte 0 (lsb) and **C/BE[3]#** applies to byte 3 (msb).

64. Ex. 2001 (PCI Spec Rev. 2.1) at 25. A PCI bus transaction consists of an address phase, followed by one or more data phases. Ex. 2001 (PCI Spec Rev. 2.1) at 25 (description of A/D[31::00]). A diagram of a PCI standard bus write transaction is set out in the portion of the PCI local bus standard below:

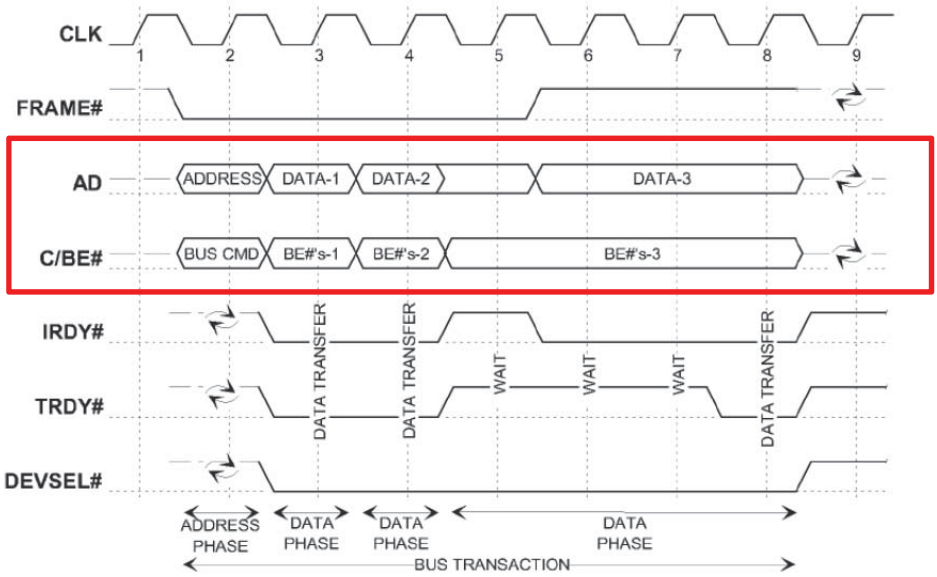


Figure 3-2: Basic Write Operation

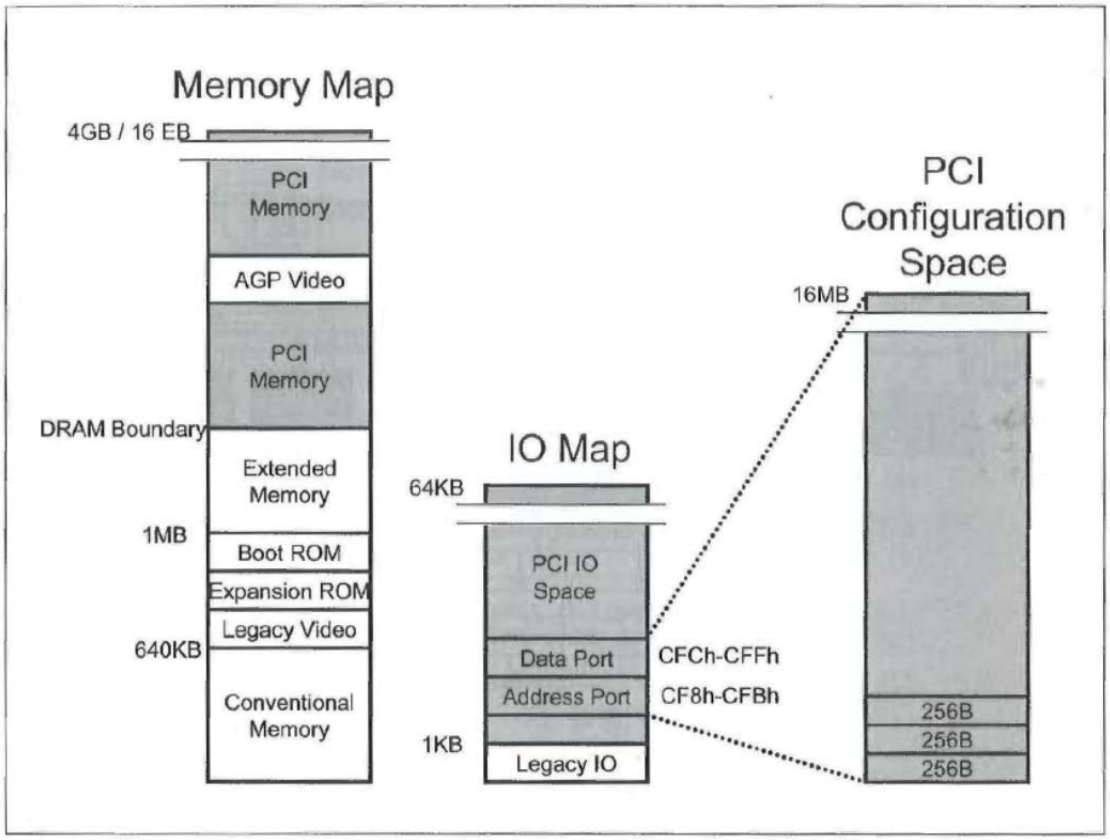
65. Ex. 2001 (PCI Spec Rev. 2.1) at 53. The PCI address and data (AD) and the bus command and byte enables (C/BE#) are highlighted. The PCI standard is based on a flat addressed hierarchical model, using a single bus arbitrator with multiple masters. Ex. 2015 at 3-5 (PCI/PCI-X book at 19-21). Flat addressing is a key component of the PCI standard because it allows the connection of multiple PCI busses through the use of transparent bridges. Ex. 2016 at 6 (Intel Nontransparent Bridge Spec at 2); Ex. 2015 at 9, 12 (PCI/PCI-X book at 76, 1143). This means that each device on a PCI bus must be mapped to a unique physical address in the processor's address space and any device on the PCI bus can read or write data directly to or from another device's address space. Ex. 2015 at 9, 12 (PCI/PCI-X book at 76 ("specific address"), 1143); Ex. 2016 at 6 (Intel Nontransparent Bridge Spec at 2). It is important to note that even minor

deviations from a standard will result in incompatible hardware and software in a designer's system.

66. The PCI standard's flat addressing model makes sense because it is a local bus designed to connect with one CPU (or multiple CPUs or processor cores) sharing a single memory and I/O address space with the CPU's local I/O devices.

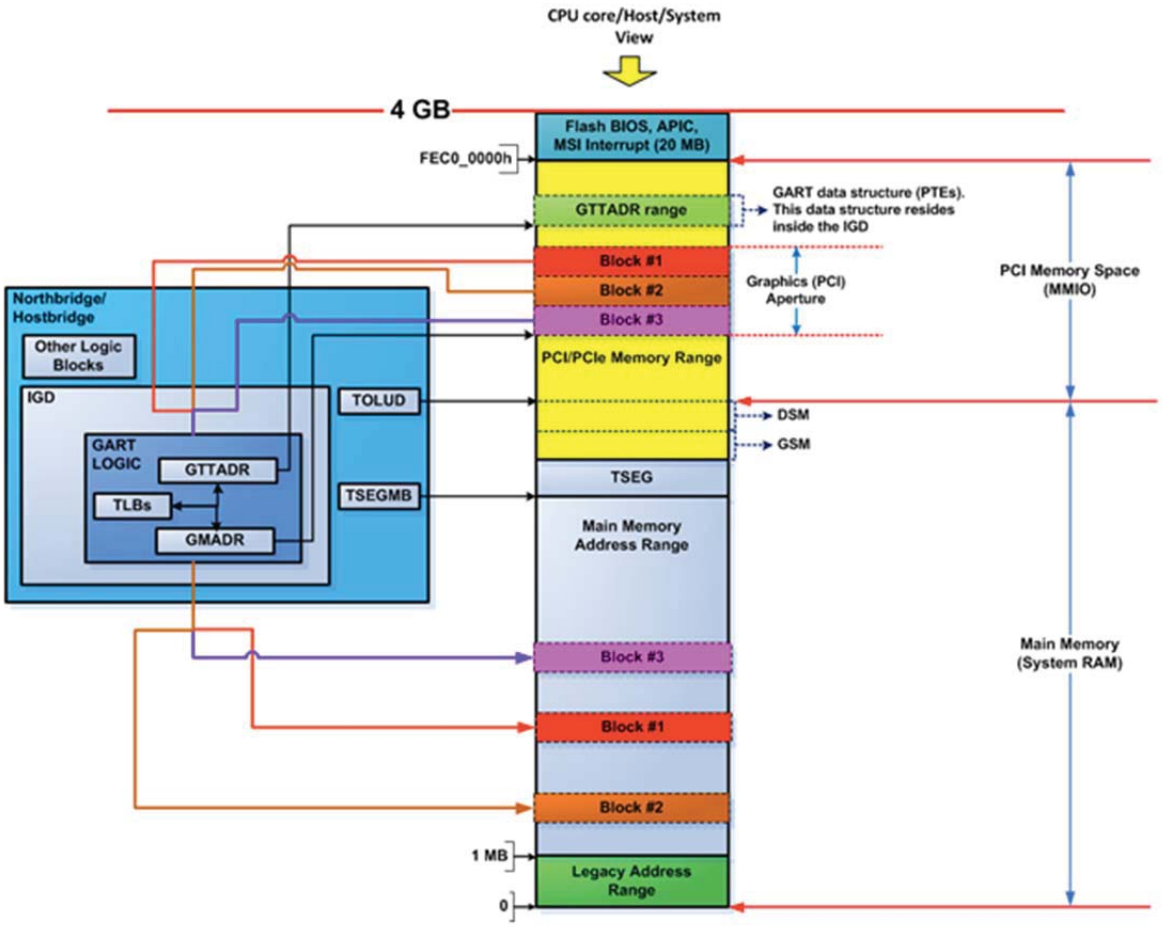
67. The PCI standard defines three physical address spaces: memory, I/O, and configuration. Ex. 2001 (PCI Spec Rev. 2.1) at 42. The PCI configuration address space is used by the PCI host to set up the PCI bus and map the PCI devices into the processor's memory address space or I/O address space. The different PCI address spaces are shown in the following figure.

Figure 1-11: Address Space Mapping



68. Ex. 2014 at 7 (PCI Express Sys. Architecture at 28). A system's memory address space is 4GB for a 32-bit address space and 16 EB (exabytes) for a 64-bit address space. PCI devices may be mapped into the system's memory address space at available address ranges, generally above the system's DRAM address range. A system's I/O address space (to the extent it exists) is smaller than the system's memory address space (64 KB in the above figure). PCI devices may be mapped into the system's I/O address space at address ranges above the legacy I/O range (1KB in the above figure).

69. The following figure illustrates an example of an Intel x86 32-bit processor's memory address map for a 32-bit or 4-GB memory address space.



70. PCI devices that are typically accessed via PCI memory bus commands are mapped into the system's memory address space and are assigned address ranges in the PCI memory space range of the system's memory address map. When the CPU issues a memory command, such as a "load" command or a "store" command on the front side bus to a memory address that corresponds to a PCI device, the memory controller, or North Bridge, generates a PCI memory

transaction on the PCI bus using the memory address assigned to the PCI device. Figures detailing the PCI transaction and TNet transaction process are attached to my declaration as Appendix A.

71. PCI devices that are accessed via PCI I/O bus commands are mapped into the system's I/O address space and are assigned address ranges in the PCI I/O address space range. When the CPU issues an I/O command on the front side bus, such as an "in" or "out" instruction, to an I/O address assigned to a PCI device, the memory controller, or North Bridge, generates a PCI I/O transaction on the PCI bus using the I/O address assigned to the PCI device.

72. These PCI address spaces uses different data formats. Ex. 2001 (PCI Spec Rev. 2.1) at 37-42 (memory and configuration transactions use DWORD addresses (double words or 32-bit increments) while I/O uses byte addresses (byte or 8-bit increments)). The PCI devices request which address space, and how much of that address space, they will be mapped into. Ex. 2001 (PCI Spec Rev. 2.1) at 42. The PCI specification recommends that PCI devices be designed to request memory-mapped address space and discourages the use of I/O address space because the I/O space is not supported in all systems and is limited and highly fragmented in PC systems. Ex. 2001 (PCI Spec Rev. 2.1) at 42.

73. The PCI specification defines specific bus transactions that use the three different PCI physical address spaces, including memory read-and-write

transactions, I/O read-and-write transactions, and configuration read-and-write transactions. Ex. 2001 (PCI Spec Rev. 2.1) at 41. These transaction types are defined by bus commands that tell the PCI components which address space is being utilized in the particular transaction. Ex. 2001 (PCI Spec Rev. 2.1) at 37. Memory transactions use memory-mapped addresses (commonly known as memory-mapped I/O) corresponding to the processor's memory address range. Ex. 2015 at 9-10 (PCI/PCI-X book at 76, 83). I/O transactions use legacy PC processor I/O address space. Configuration cycles use device identification and bus location to address devices. The PCI specification also defines a special cycle and an interrupt acknowledge transaction that does not utilize an address. Ex. 2001 (PCI Spec Rev. 2.1) at 37.

74. Memory and I/O transactions are both used to read and write data between the host processor's memory and I/O devices attached to the PCI bus. Ex. 2001 (PCI Spec Rev. 2.1) at 37-38. Configuration read and writes are used to initialize and map the devices attached to the PCI bus into either memory or I/O address space on bus setup. Ex. 2001 (PCI Spec Rev. 2.1) at 38-39. The PCI transaction types define different address spaces and different address formats for addressing PCI components on a PCI bus. Ex. 2001 (PCI Spec Rev. 2.1) at 41.

75. Every device on the PCI bus is responsible for decoding the PCI address when a PCI bus transaction is detected. Ex. 2001 (PCI Spec Rev. 2.1) at

42. This requires that a PCI standard address be decodable by any PCI-compliant device. In addition, the PCI standard does not allow byte lane swapping on the address and data lines. Ex. 2001 (PCI Spec Rev. 2.1) at 45. Instead, all address bits must be maintained in the proper order for address decoding purposes. Ex. 2001 (PCI Spec Rev. 2.1) at 45.

76. Without the PCI bus command specific to the transaction type, and a corresponding physical PCI address in the correct format and to the correct address space, a transaction cannot be a PCI standard bus transaction. In fact, all PCI devices must support PCI configuration commands and the configuration address space. Further, a PCI bus master must be able to support any PCI-compliant component, whether it uses memory address space or I/O address space, using at least the basic I/O and memory read-and-write commands.

77. PCI bus commands are set out in the following table from the PCI 2.1 standard:

C/BE[3::0]#	Command Type
0000	Interrupt Acknowledge
0001	Special Cycle
0010	I/O Read
0011	I/O Write
0100	Reserved
0101	Reserved
0110	Memory Read
0111	Memory Write
1000	Reserved
1001	Reserved
1010	Configuration Read
1011	Configuration Write
1100	Memory Read Multiple
1101	Dual Address Cycle
1110	Memory Read Line
1111	Memory Write and Invalidate

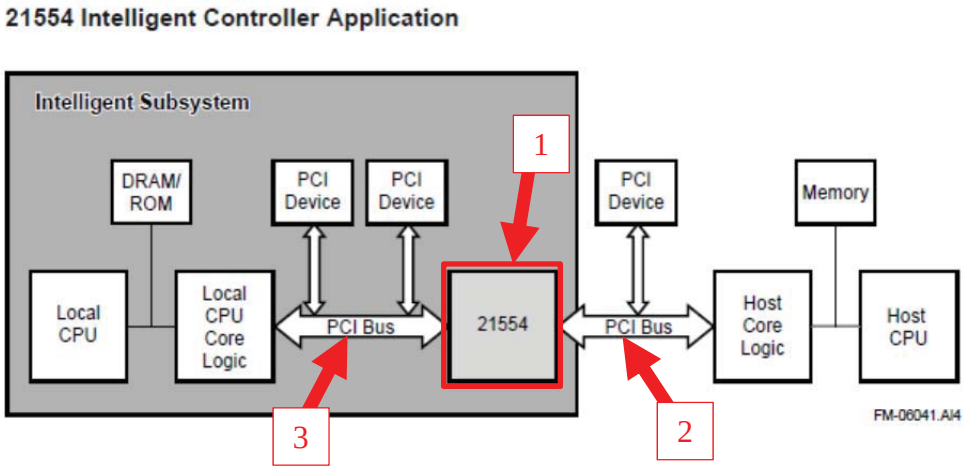
Ex. 2001 (PCI Spec Rev. 2.1) at 37-39.

78. As a local bus protocol, the PCI standard is not designed to, and cannot work with, multiple processor nodes implementing separate address spaces. Ex. 2016 at 6 (Intel Nontransparent Bridge spec at 2). A special bridge device is required to connect PCI busses between address spaces. Ex. 2016 at 6 (Intel Nontransparent Bridge spec at 2). These bridge devices effectively terminate one PCI bus and create another PCI bus with a separate address space. Ex. 2016 at 6 (Intel Nontransparent Bridge spec at 2).

79. The PCI local bus standard will not work in networked multi-node systems for at least a couple of reasons. First, PCI implements a flat address space

and requires each device to decode the addresses on the bus, while in a multi-node environment each processor node will have its own separate address space. Ex. 2015 at 9, 12 (PCI Book at 83, 1143); Ex. 2017 at 3-4 (VITA Journal at 6-7). As a result, there will be address conflicts. Ex. 2017 at 3-4 (VITA Journal at 6-7). Second, a multi-node environment will cause issues with the PCI plug-and-play functionality because each processor and memory system on each node will have a memory controller that attempts to set up the PCI bus separately, resulting in address collisions. Ex. 2016 at 6 (Intel Nontransparent Bridge spec at 2); Ex. 2017 at 3-4 (VITA Journal at 6-7). Non-transparent or embedded bridges must be used to connect separate processors over a PCI bus. Ex. 2017 at 3-4 (VITA Journal at 6-7). Non-transparent bridges effectively terminate both PCI busses. These non-transparent bridges do not pass PCI transactions and the PCI bus arbitrators for the respective PCI devices are not aware of the PCI devices on the far side of a non-transparent bridge. Instead non-transparent bridges use processors that are aware of the address space and devices on each side to generate new PCI bus transactions appropriate for the target bus by creating PCI addresses tailored to that PCI buss's address space.

80. The following figure is from the Intel 21554 PCI-to-PCI Bridge for Embedded Applications and provides an example of how a non-transparent PCI bridge might be connected.



81. Ex. 2016 at 6 (Intel Nontransparent Bridge Spec at 2). The 21554 (1) is the non-transparent PCI-to-PCI bridge. The primary PCI bus (2) on the main computer system is managed by the Host Core Logic and Host CPU. The secondary PCI bus (3) is managed by the Local CPU and Local CPU Core Logic on the Intelligent Subsystem. The primary PCI bus will be mapped into the Host CPU's memory or I/O address map. The Host CPU will map the non-transparent bridge's (1) primary PCI bus side into its memory address space, but neither the Local CPU or any PCI device on the secondary PCI bus (3) will be visible to the Host CPU. Instead, a custom driver for the non-transparent bridge and its secondary PCI connections is required.

82. The secondary PCI bus (3) is managed by the Local CPU on the intelligent subsystem. The PCI devices, including the non-transparent bridge's (1) secondary PCI bus on the secondary PCI bus (3) are mapped into the Local CPU's memory or I/O address space.

83. When a transaction is initiated by the Host CPU that addresses the Intelligent Subsystem, the Host CPU issues a command to the address range in the Host CPU's address space corresponding to the non-transparent bridge's (1) primary PCI bus side (2). The non-transparent bridge (1) generates a new PCI bus transaction on the secondary PCI bus using a new PCI address, and possibly a new PCI bus command based on information known only to it about the Intelligent Subsystem and the Intelligent Subsystem's PCI address space.

4. '873 Patent Solves the Problem of Interconnecting ACMs with a PCI Standard Bus and PCI Components.

84. In designing the solution to the problem of a modularized computer with access to peripherals at a speed sufficient to keep up with ever-increasing processor speeds, several requirements were important to consider. Particularly, at the time of the invention, peripherals were primarily connected to processor and memory through a PCI bus. Ex. 2001 (PCI Spec. Rev. 2.1) at 3; Ex. 2013 at 9 (CD&O at 687). To maintain the communication standard and ensure widespread acceptance, it was important to communicate with PCI peripherals without requiring modification of drivers and peripherals. Ex. 1001 ('873 patent) at 4:50-58; Ex. 2012 at 20 (CA at 531). Further, there could not be a performance penalty because slow communication with peripherals would ensure that the system would not be widely accepted, as one of the primary design considerations in computer

architecture is speed. Ex. 1001 ('873 patent) at 4:50-58. Finally, it was important to have hardware that had a small form factor and was cost effective in combination with a communication protocol that was cable-friendly. Ex. 1001 ('873 patent) at 3:32-67; 5:40-5:62.

85. The inventor recognized that a serial communication scheme would allow for reliable, fast communication. In fact, communicating the PCI transactions over serial channels, rather than the standard parallel bus used for PCI, resulted in faster communication with no performance penalty. Ex. 1001 ('873 patent) at 3:32-67; 4:50-57. The lower pin count of the invention makes the connector more reliable and less expensive. Ex. 1001 ('873 patent) at 5:40-62. Finally, an additional benefit of using the serial communication line allows for a longer communication path to the peripheral devices, which is very limited for traditional PCI. Ex. 1001 ('873 patent) at 5:40-62. An example of LVDS serial channels for carrying serialized PCI bus transaction described the '873 patent is set out below.

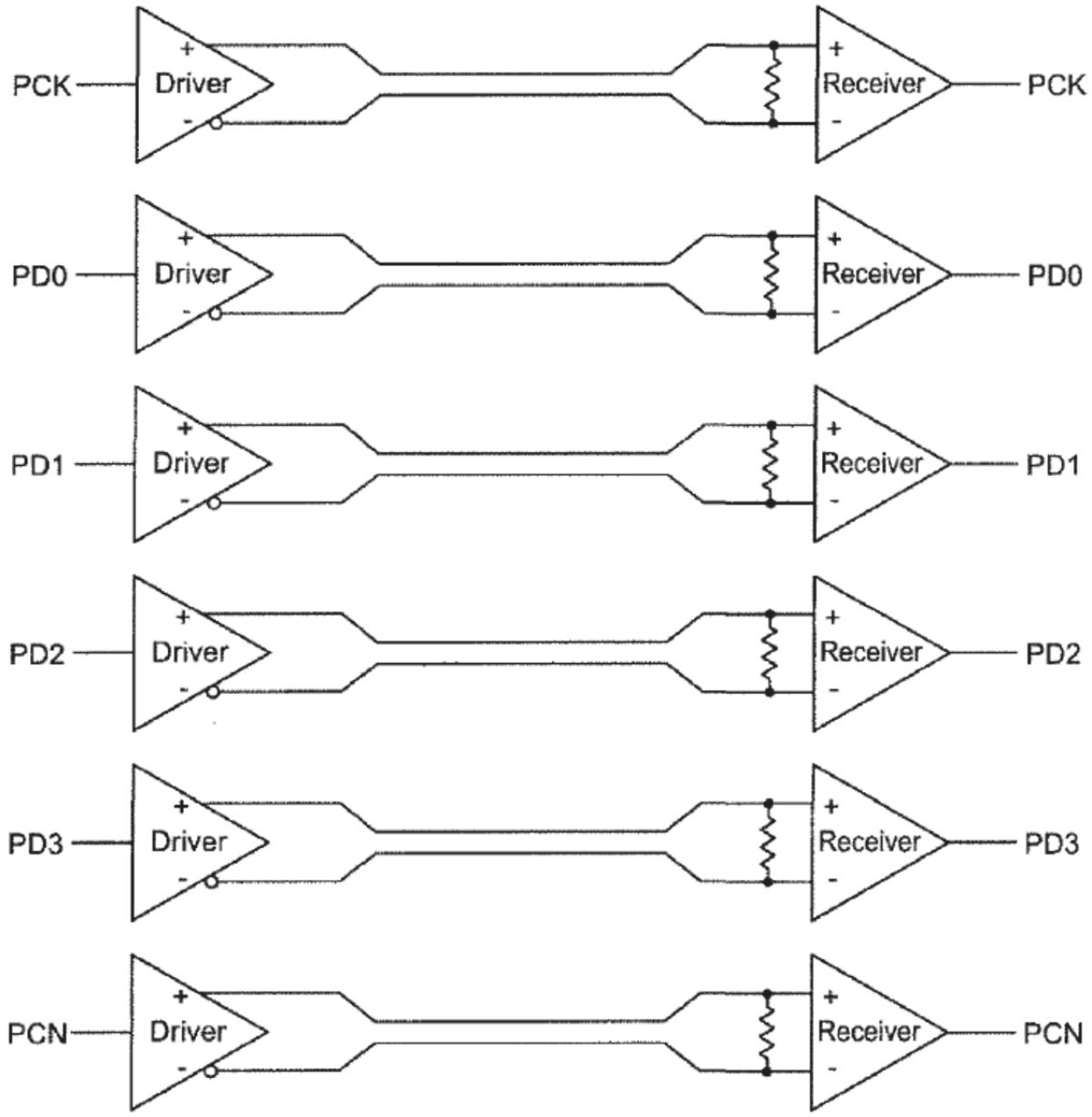


FIGURE 15

86. Ex. 1001 ('873 patent) at FIG. 15. Examples of serialized PCI bus transactions described in the '873 patent are set out in Figures 13 and 14 of the '873:

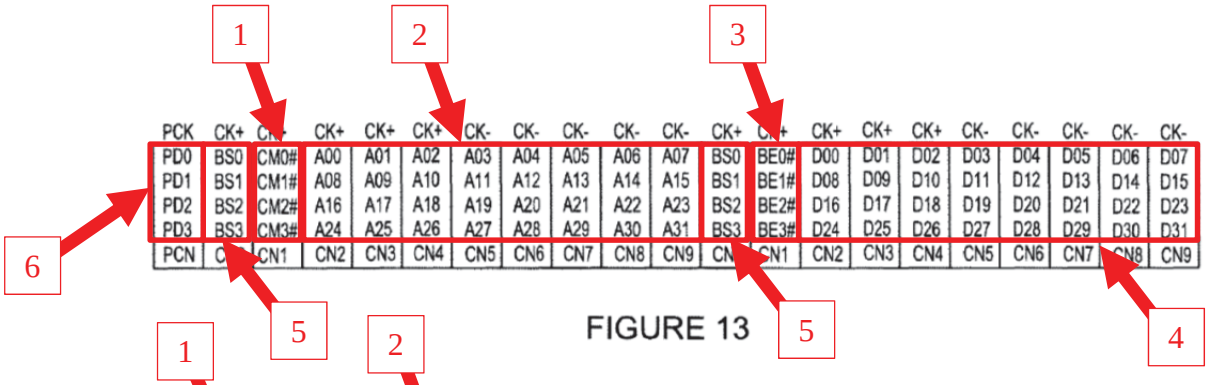


FIGURE 13

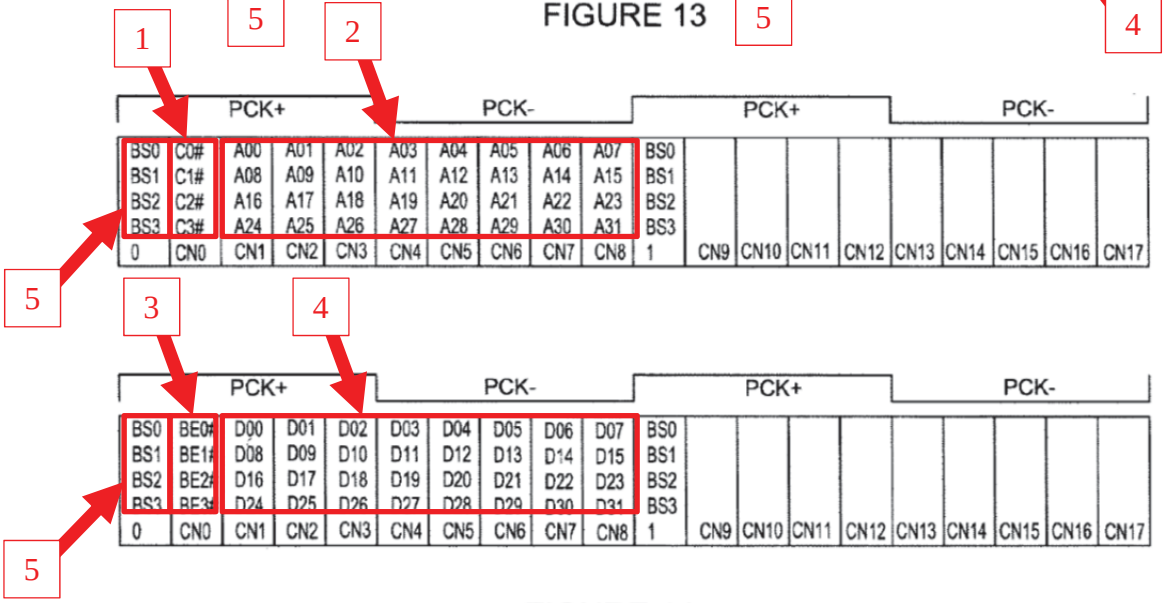


FIGURE 14

87. Ex. 1001 ('873 patent) at FIGs. 13, 14; 20:27-21:31. I have annotated Figures 13 and 14 to show the different serial bits I discuss below. The CM0#-CM3# in Figure 13 and C0#-C3# (1) in Figure 14 represent the 4 bits of PCI standard bus commands indicating the type of transaction C/BE# [3::0]. Ex. 1001 ('873 patent) at FIGs. 13, 14; 20:34-35; 20:44-47; 20:61-21:19. The A00-A31 in Figures 13 and 14 (2) represent the 32 bits of PCI standard addresses A[31::0]. Ex. 1001 ('873 patent) at FIGs. 13, 14; 20:30-31; 20:61-21:19. The BE0#-BE3# in Figures 13 and 14 (3) are the 4 bits of PCI standard byte enables asserted on the bus command lines during the data phase of a PCI standard bus transaction C/BE#

[3::0]. Ex. 1001 (‘873 patent) at FIGs. 13, 14; 20:35-36; 20:44-47; 20:61-21:19. The D00-D31 in Figures 13 and 14 (4) represent 32 bits of PCI standard data D[31::0]. Ex. 1001 (‘873 patent) at FIGs. 13, 14; 20:31-32; 20:61-21:19. The BS0-BS3 in Figures 13 and 14 (5) are “bus status” bits added by the inventor to facilitate serialized communication and form 10 bit packets, which may include additional PCI-related information like the FRAME#, IRDY#, and TRDY# signals. Ex. 1001 (‘873 patent) at FIGs. 13, 14; 20:31-32; 20:37-39; 20:48-52. The PD0-PD3 in Figure 13 (6) are the 4 serial data channels that transmit the serialized PCI bus transaction. PCK is a variable clock that can operate at higher frequencies than the PCI bus clock and is asynchronous with the PCI bus clock. Ex. 1001 (‘873 patent) at 15:46-63. PCN is a control signal line that can include CPU to South Bridge signaling, such as CPU interrupts. Ex. 1001 (‘873 patent) at 20:39-40; 20:53-60.

88. The ‘873 patent discusses serial lines and serial packet protocols and uses LVDS, IEEE 1394 (FireWire) and Universal Serial Bus (USB) as examples. Ex. 1001 (‘873 patent) at 22:23-27. For example, LVDS encodes serial bits as voltage differences onto opposed lines, such as in Figure 15 of the ‘873 patent. IEEE 1394 uses Data strobe encoding and clock synchronization with a phase-locked loop. Consistent with this, the ‘873 patent states that when “low voltage

digital signal" or "LVDS" is used in the patent, it not limited to any particular technology. Ex. 1001 ('873 patent) at 4:1-4.

89. The solution in the '873 patent also maintains the full PCI standard transaction types, address data format, address space, and data format. The benefit is that it is transparent to software, meaning no new drivers are needed to ensure the peripherals work on the system. Ex. 1001 ('873 patent) at 4:50-58. To maintain software transparency, it is critical to maintain the same bus commands and address spaces. Ex. 2017 at 3 (PCI Express Sys. Architecture at 10). Additionally, there is interoperability with existing peripherals, including the connectors and protocols. The system as described has the added benefit that it is plug-and-play because no special setup protocols or new device drivers are needed. Finally, the system is transportable because the serialized PCI transactions have the same PCI meaning on all chassis and to all endpoints.

90. The claims implement the solution using low voltage differential signal (LVDS) multibit serial channels to transmit encoded serial bit streams of PCI bus transactions. Ex. 1001 ('873 patent) at claim 54. The specification describes serially encoding PCI bus transactions by taking PCI bus signals for command, byte enable, address, and data, and encoding them as bits on one or more serial channels. Ex. 1001 ('873 patent) at FIGs. 13, 14, and 15; 20:27-47. That PCI bus commands and addresses are maintained allows software

transparency. Ex. 1001 ('873) at 4:54-56; Ex. 2014 at 3 (PCI Express Sys. Architecture at 10). That PCI bus commands, addresses, and data are maintained on the serial channels allows transparent decoding and bridging, which is a primary goal of the PCI standard for bus interconnects. Ex. 1001 ('873 patent) at 4:56-58; Ex. 2016 at 6 (Intel nontransparent bridge spec at 2); Ex. 2015 at 10, 12 (PCI/PCI-X book at 76, 1143).

5. PCI Express

91. In late 2001, Intel and the PCI-SIG announced they were developing a new local bus specification, initially called 3GIO, or Third Generation IO Interconnect. 3GIO was later renamed PCI Express. PCI Express is a serial communication protocol using 1, 2, 4, 8 or more low voltage digital signal channels with 8b/10b line encoding for communicating with a CPU's local I/O peripheral devices. PCI Express maintains PCI's bus commands and three flat physical address spaces which allows backwards compatibility with PCI devices because no changes to software or drivers are required. Ex. 2017 at 3 (PCI Express Sys. Architecture at 10). PCI Express's backwards compatibility requirements with PCI are set out in the PCI Express standard:

❑ **PCI compatible software model:**

- Ability to enumerate and configure PCI Express hardware using PCI system configuration software implementations with no modifications
- Ability to boot existing operating systems with no modifications
- Ability to support existing I/O device drivers with no modifications
- Ability to configure/enable new PCI Express functionality by adopting the PCI configuration paradigm

Ex. 2019 at 38 (PCI Express Spec v3.0 at 38).

92. To do this PCI Express uses serialized PCI bus transactions with the same CPU operations, bus commands, byte enables, and memory, I/O, and configuration addresses as PCI. Ex. 2020 at 3 (RapidIO Whitepaper at 3); Ex. 2019 at 30, 32-33, 37, 48, 69 (PCI Express Spec v3.0 at 30, 32-33, 37, 48, 69). The following figure shows the PCI Express bus transaction types:

Table 2-1: Transaction Types for Different Address Spaces

Address Space	Transaction Types	Basic Usage
Memory	Read Write	Transfer data to/from a memory-mapped location
I/O	Read Write	Transfer data to/from an I/O-mapped location
Configuration	Read Write	Device Function configuration/setup
Message	Baseline (including Vendor–defined)	From event signaling mechanism to general purpose messaging

Ex. 2019 at 54 (PCI Express Spec v3.0 at 54). The Memory, I/O, and Configuration address spaces and transactions include the PCI standard address spaces and transactions.

93. The following figure shows an example PCI Express packet header that encodes the PCI standard transaction type (1), the PCI standard byte enables (2), and the PCI standard address (3), in this case a PCI memory address mapped to the CPU's physical memory space:

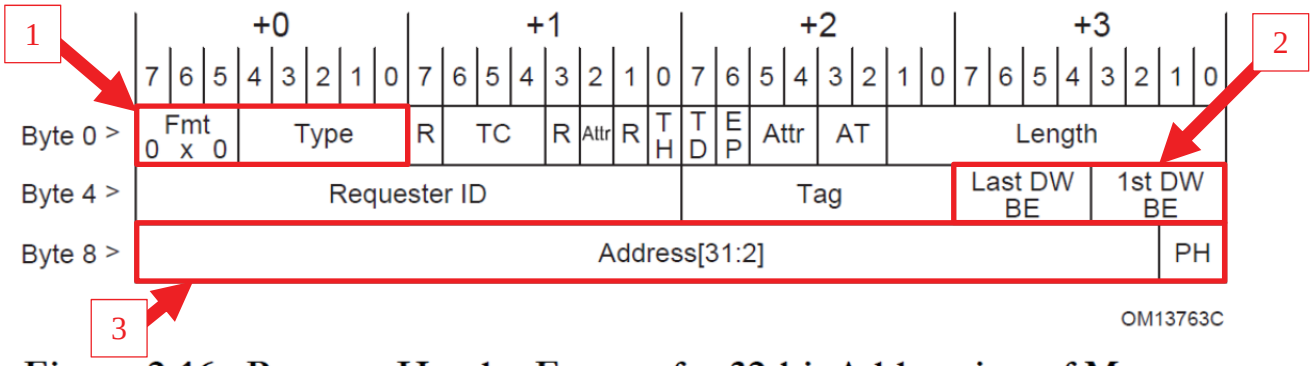
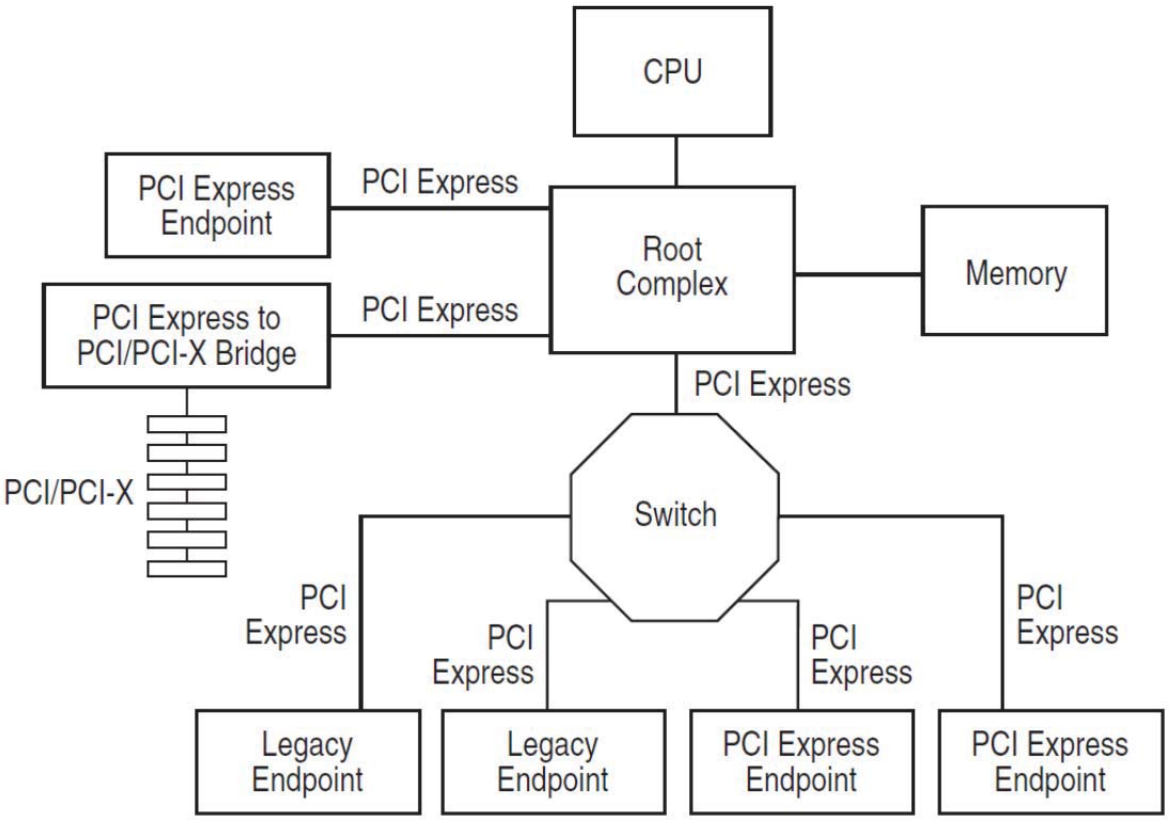


Figure 2-16: Request Header Format for 32-bit Addressing of Memory

Ex. 2019 at 78 (PCI Express Spec v3.0 at 78).

94. An example of the layout of a computer-system using PCI Express is set out below.



95. Ex. 2019 at 41 (PCI Express Spec v3.0 at 41). In PCI Express, the Root Complex takes the place of the memory controller/North Bridge. Further, instead of a bus, PCI Express uses point-to-point links, like that shown in Figure 18 of the '873 patent. Ex. 1001 ('873 patent) at Fig. 18. The PCI Express Root Complex addresses PCI Express memory, I/O, and configuration read-and-write transactions using the PCI standard addresses for the respective memory space.

96. PCI Express encodes the parallel PCI bus commands, byte enables, and address signals into bits and orders them on the PCI Express low voltage

digital signal serial bit channels. The following figures from the PCI Express specification show how this bit ordering of PCI bus command was accomplished.

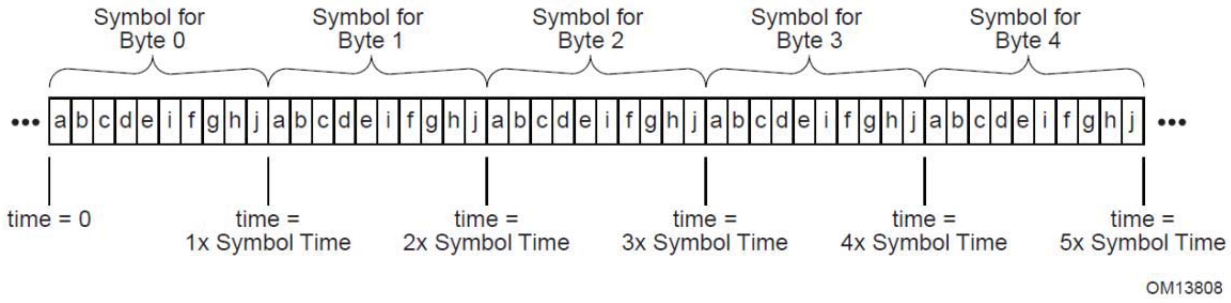


Figure 4-3: Bit Transmission Order on Physical Lanes - x1 Example

97. Ex. 2019 at 193 (PCI Express Spec v3.0 at 193). The above figure shows the bytes of a PCI Express packet, including the PCI transaction type, PCI byte enables, and PCI address encoded with 8b/10b encoding into 10-bit packets and being sent serially over a single differential channel.

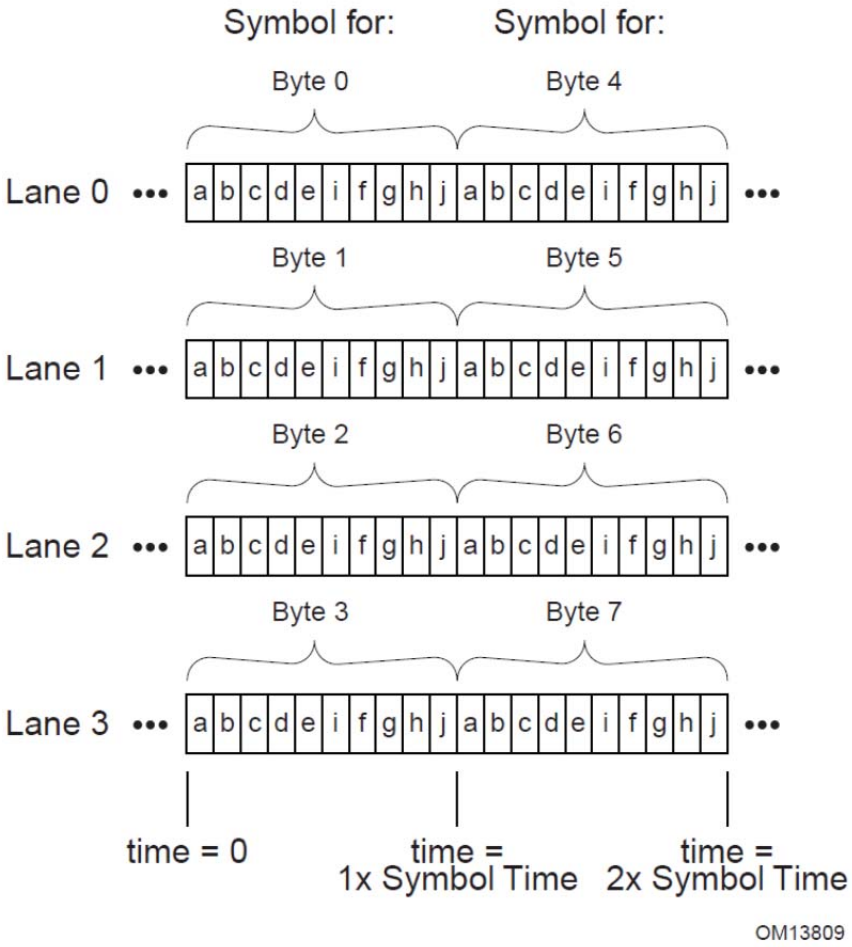


Figure 4-4: Bit Transmission Order on Physical Lanes - x4 Example

98. Ex. 2019 at 193 (PCI Express Spec v3.0 at 193). The above figure shows the bytes of a PCI Express packet, including the PCI transaction type, PCI byte enables, and PCI address encoded with 8b/10b encoding into 10-bit packets and ordered to be sent serially over four differential channels.

99. I understand that ACQIS is asserting the '873 patent against EMC products implementing PCI Express busses. I also understand that ACQIS has asserted patents related to the '873 patent against other products implementing PCI

Express busses in the past. In my opinion, this is consistent with the Board's construction of "PCI bus transaction" as a "PCI standard bus transaction," which I address more below.

6. Horst's TNet Disclosure Addresses a Different Issue – Interconnecting Remote Processors

100. The TNet solution, discussed in the Horst reference, was not solving the same problem as that encountered by Dr. Chu in the '873 patent. Rather than interconnecting a processor with local peripherals, TNet seeks to interconnect multiple independent processor and memory system nodes. See Ex. 1011 (Horst) at 1. Horst states that TNet provides for distributed memory multi-computing and that provides common software and hardware services to processor and I/O nodes. Ex. 1011 (Horst) at 1. In other words, TNet seeks to solve a different problem than PCI, PCI Express, and the claimed inventions. PCI, PCI Express, and the claimed inventions seek to connect a single CPU or CPU node to its local I/O components. See discussion above. TNet seeks to connect hundreds of remote processor-memory nodes, each executing its own operating system, with one another and thousands of distributed I/O resources. Ex. 1011 (Horst) at 1.

101. TNet was successful at its intended goal of connecting distributed processor and memory nodes. The modern Infiniband standard inherited many features from TNet. Infiniband is a high speed processor and memory interconnect

fabric. I used the Infiniband protocol in creating the L-CSC supercomputer to connect multiple separate processing and memory resources for parallel computing.

102. TNet's design, as a different solution to a different problem, explains some of the key differences between it and the PCI standard. The author of TNet was looking to develop a system capable of interconnecting hundreds of processor nodes (each with its own memory space) to share memory and processing capabilities while allowing the processors to share I/O resources. Ex. 1011 (Horst) at 1. In order to accomplish this goal, the author evaluated existing communications standards but "found nothing that met all the requirements." Ex. 1011 (Horst) at 2. PCI was an existing bus standard at the time of the Horst paper. "This forced [the author] into the difficult job of designing a completely new network." Ex. 1011 (Horst) at 2. As a result, the author specifically chose to avoid the PCI standard bus transaction protocol and any existing protocol in favor of a "completely new network" using a low overhead generic peer-to-peer protocol. See Ex. 1011 (Horst) at 2, 5, Figs. 2, 5, 9.

103. This makes sense because the PCI standard uses a single processor node address space to address transactions. Ex. 2016 at 6 (Intel Nontransparent Bridge Spec at 2); Ex. 2015 at 9, 12 (PCI/PCI-X Book at 76, 1143); Ex. 2020 at 3 (RapidIO Whitepaper at 3) (discussing PCI and PCI Express, "When you try to

build a multi-processor interconnect out of PCI, you, of necessity, must step beyond the base PCI specification and create new mechanisms to map address spaces and device identifiers among multiple host or root processors. To date none of the proposed mechanisms to do this — Advanced Switching (AS), Non-transparent Bridging (NTB) or Multi-Root – I/O Virtualization (MR-IOV) — have been commercially successful nor do they support arbitrary topologies."). TNet's goal of interconnecting hundreds of processor nodes, each with its own memory address space, using PCI standard bus transactions would cause address conflicts and confusion because each processor node has the same address range. Ex. 2016 at 6 (Intel Nontransparent Bridge Spec at 2); Ex. 2017 at 3-4 (VITA Journal at 6-7). Horst's figure 9 shows a diagram of the TNet architecture.

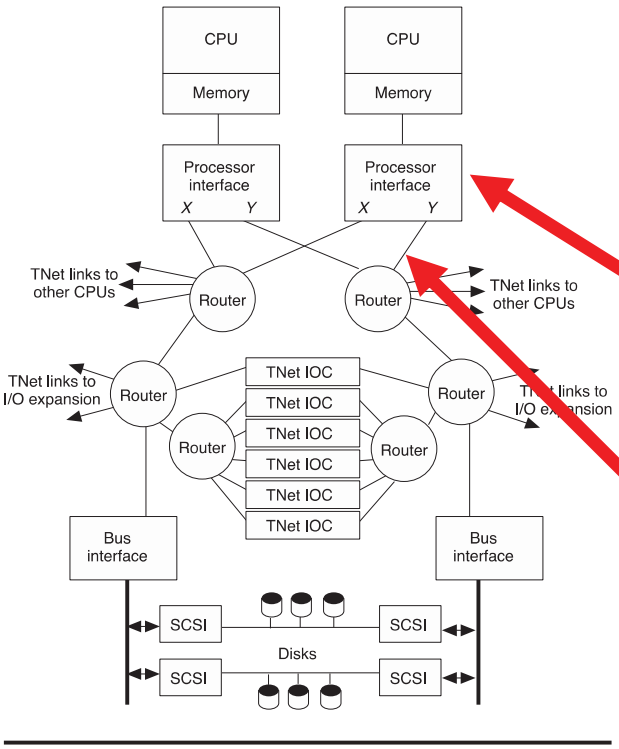


Figure 9. Typical system architecture using TNet.

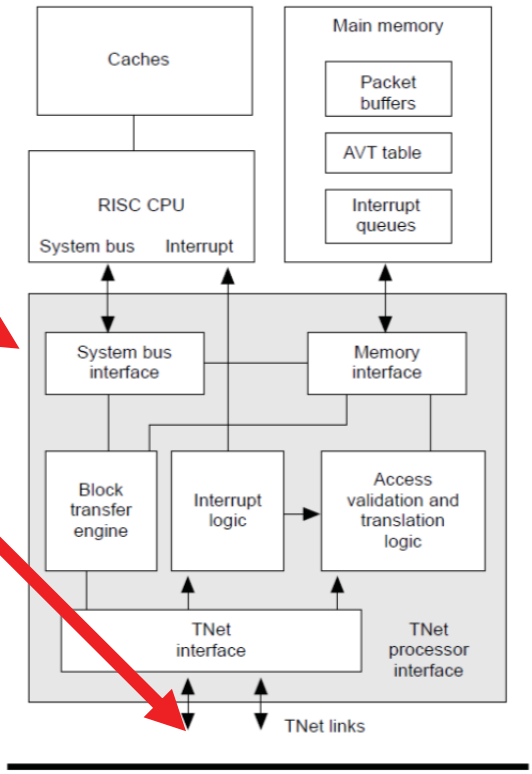


Figure 7. TNet processor interface.

Ex. 1011 (Horst) at 6, 8, Fig. 7, 9. Note that the TNet architecture replaces the local bus architecture with what is commonly known as a memory fabric, interconnecting multiple separate CPU's and main memory using a node ID routed peer-to-peer network. TNet does this by replacing the north bridge with the TNet processor interface. Ex. 1011 (Horst) Figure 7. Figure 9 also shows that bus interface connections are remote to the CPUs; in Figure 9, this is a SCSI bus. Figure 2 from Horst is a block diagram of the new TNet network.

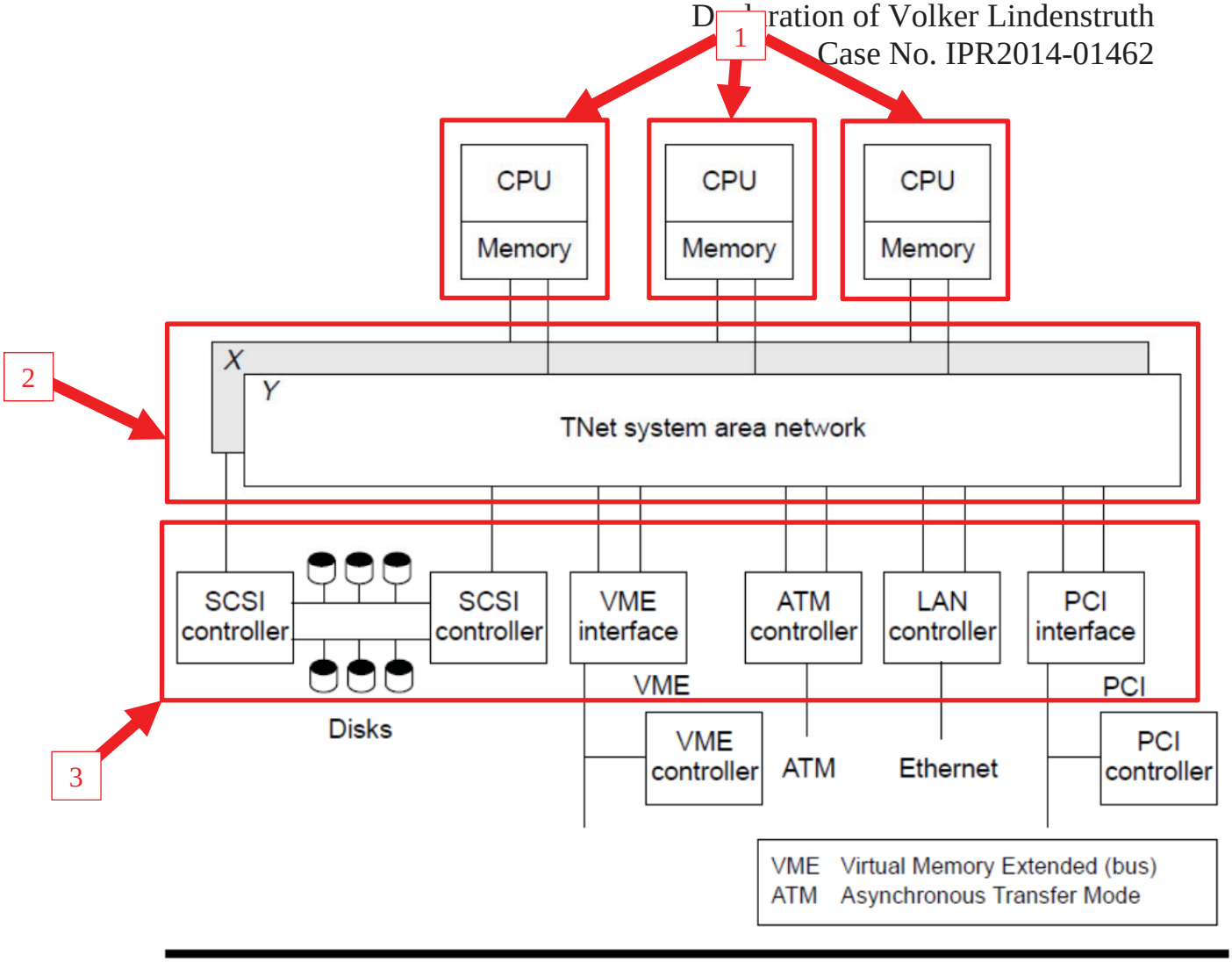
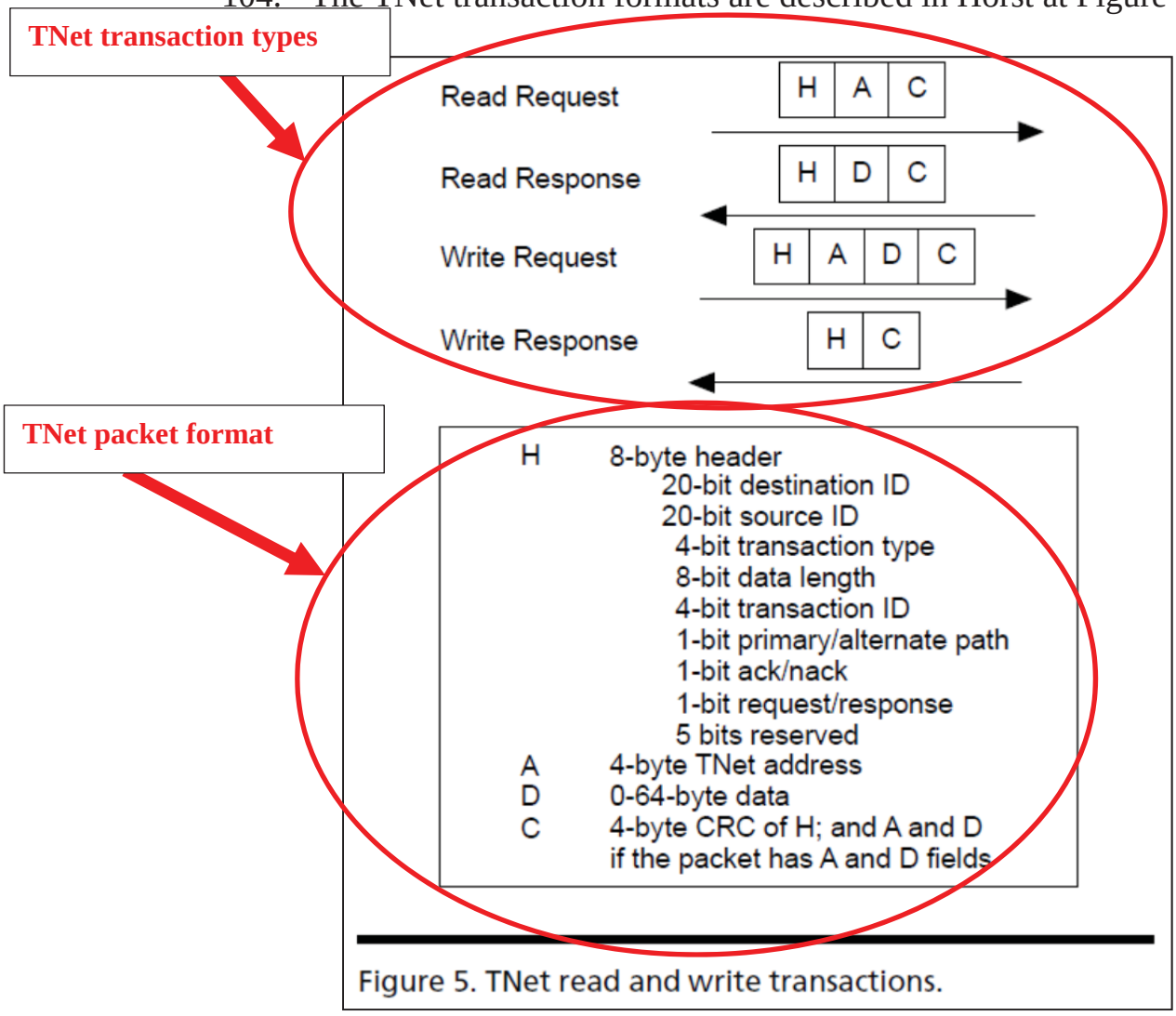


Figure 2. CPU cluster connected with the TNet system area network.

Ex. 1011 (Horst) at Fig. 2. As stated, TNet contemplates interconnecting separate CPU nodes and main memory (e.g. 1) via TNet (2) through TNet Processor Interface (see Ex. 1011 (Horst) at Fig. 7). The TNet Processor Interface generates TNet transactions from CPU frontside bus transactions, including applying a destination ID, and TNet transaction type (see Ex. 1011 (Horst) at Fig. 5), and sends the TNet transaction over the TNet link to the TNet network (2). The TNet network (2) interfaces with multiple different TNet bus interface controllers (see

Ex. 1011 (Horst) at Fig. 8) (3). Only after a TNet transaction arrives at TNet bus controller (3) does the intelligent ASIC in the TNet bus interface use the TNet transaction to generate a bus transaction.

104. The TNet transaction formats are described in Horst at Figure 5.



Ex. 1011 (Horst) at 5, Fig. 5. TNet packets use 4-byte virtual TNet addresses to address memory in TNet nodes. Ex. 1011 (Horst) at 5, Fig. 5. TNet addresses

serve as virtual I/O addresses to consolidate scattered physical memory. Ex. 1011 (Horst) at 1-2.

105. The "completely new network" described by Horst requires that new specially-designed intelligent TNet processor and I/O nodes for each type of processor and bus be created. Ex. 1011 (Horst) at 6-7, Figs. 7, 8. While the Horst paper states that the TNet bus interfaces are driven by application-specific integrated circuits (ASICs) that generate and translate TNet transactions, there is no disclosure about how those ASICs are configured. Ex. 1011 (Horst) at 6-7, Figs. 7, 8. In addition to requiring new TNet bus interfaces for each bus, new driver software for each new TNet bus interface would be required to allow a processor to communicate with a TNet bus node using TNet transactions. Ex. 1011 (Horst) at 6-7, Figs. 7, 8 ("Different versions of the bus interface logic support [different busses]"). The Horst paper provides no disclosure about how these drivers operate either. As a result, there is no disclosure about how the TNet transactions actually interoperate with the TNet nodes.

106. The '873 patent's reliance on serialized PCI standard bus transactions, rather than other existing serial protocol, avoids the requirement imposed by TNet of designing new bus interface devices and programming new device drivers. Designing new bus interfaces and programming new device drivers represents a significant expense and serves as a barrier to adoption and commercial success.

Ex. 2012 at 20 (CA at 531). In 2002, the PCI-SIG adopted the same solution as taught by the '873 patent when it issued the PCI Express revision 1.0 standard based on serialized PCI standard bus transactions. Rather than creating an entirely new protocol like TNet and require new hardware and software, the PCI Express's standard uses serialized PCI standard bus transactions, which allows PCI Express to be backwards compatible with PCI software, PCI drivers, and PCI devices. Thus, PCI Express did not require any new software or devices to work with the existing universe of PCI devices.

107. The PCI standard also teaches against implementing a system like TNet with multiple virtual address spaces. Unlike the TNet system, that requires new intelligent bus interfaces and new drivers, one of the primary goals of the PCI standard was to allow the use of transparent PCI-to-PCI bridges. Ex. 2015 at 12 (PCI/PCI-X book at 1143); Ex 2016 at 6 (Intel Nontransparent Bridge Spec at 2). PCI standard bridges are said to be "transparent" because they require no driver software to translate or route transactions. Ex. 2015 at 12 (PCI/PCI-X book at 1143); Ex 2016 at 6 (Intel Nontransparent Bridge Spec at 2). Transparent PCI bridges are made possible by the PCI standard addresses, which exist in a flat physical address space (e.g., each device on a PCI bus has a unique physical address addressed by either the host CPU's physical memory address space or

physical I/O address space). Ex. 2001 (PCI Spec Rev. 2.1) at 42; Ex. 2015 at 12 (PCI/PCI-X book at 1143); Ex. 2014 at 7 (PCI Express Sys. Architecture at 28).

108. TNet, on the other hand, is a "completely new network" that purposefully avoids PCI's memory-mapped I/O and physical addressing scheme and, instead ,uses virtual addressing, address translation, and 20-bit source IDs to route TNet transactions. Ex. 1011 (Horst) at 2, 5, 6-7, Figs. 7, 8.

109. Horst's Figure 7, below, shows the TNet processor interface, which connects to the processor (1) via the system bus (2) to the TNet processor interface (3). The TNet processor interface (3) contains the access validation and translation logic (4), which translates TNet virtual addresses into physical addresses. The TNet interface (5) sends and receives TNet packets over the TNet links (6). The TNet processor bridge is the only processor interface disclosed in Horst that is connected to low voltage serial data channels.

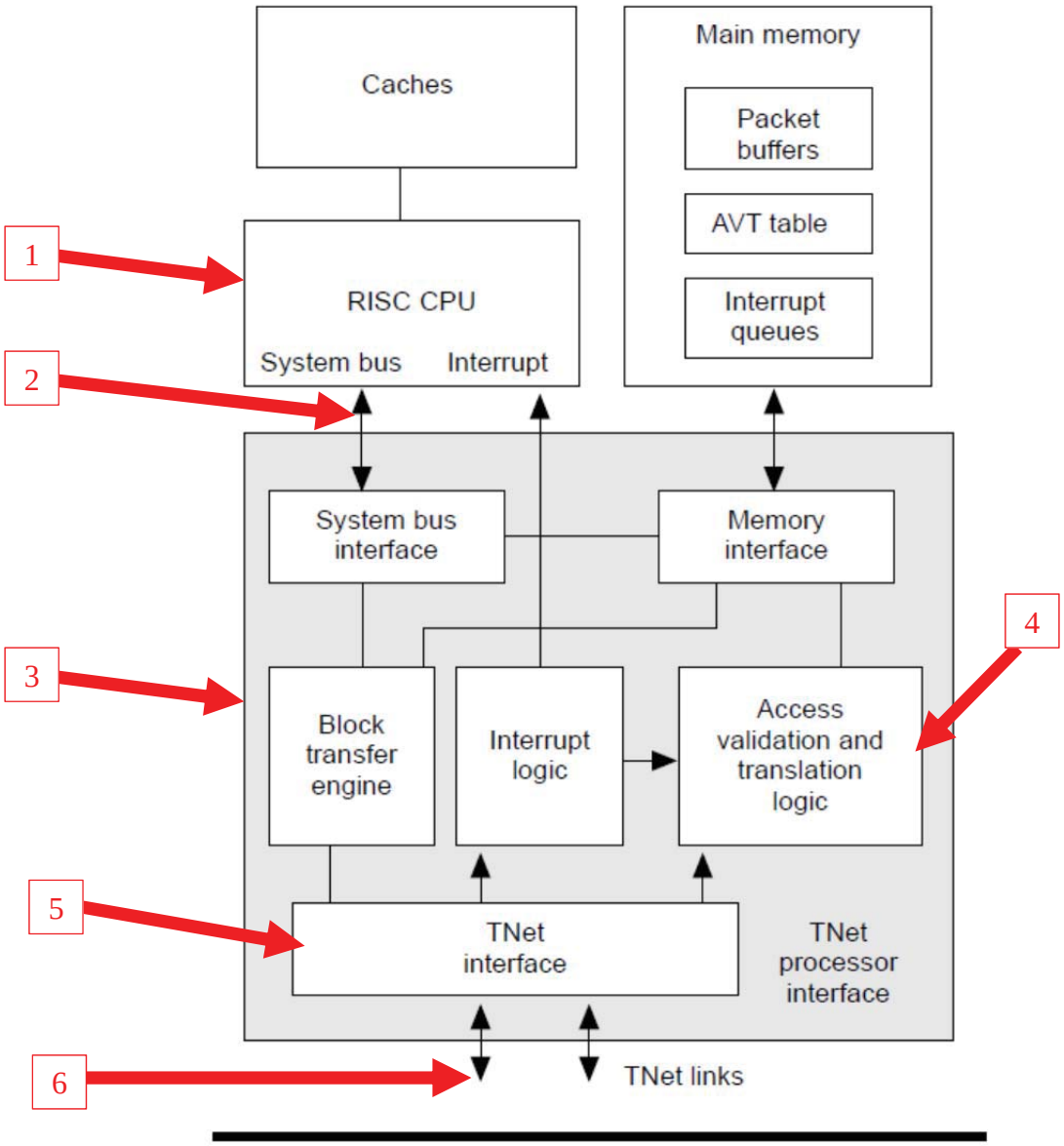


Figure 7. TNet processor interface.

110. Ex. 1011 (Horst) at Fig. 7. It is important to note that the Horst paper does not describe the process for a CPU writing data through the TNet processor interface, only for receipt of a TNet packet. Ex. 1011 (Horst) at 6-7. As a result, Horst does not disclose which of the several different ways a processor can initiate

a system bus transaction is used to communicate with the TNet processor interface in order to cause it to communicate over the TNet links. However, the Horst paper does state that each transfer has its own address validation and translation entry. Ex. 1011 (Horst) at 6. This appears to indicate that each TNet transaction uses a unique virtual address. In other words, TNet transactions to the same node and physical address could have, and apparently did have, different TNet virtual addresses. This further indicates that, in addition to TNet addresses being virtual, and therefore not reversible, TNet virtual addresses also vary on a transaction-by-transaction basis, and are not reversible for this additional reason. Instead, TNet virtual addresses appear to be created on an as-needed and arbitrary basis for each transaction.

111. Horst's Figure 8, below shows the TNet bus interface (1), which connects the TNet network to standard busses. Ex. 1011 (Horst) at 7. The TNet bus interface receives TNet transactions over the TNet link (2), which are depacketized (3) and sent to the Bus interface (4). Ex. 1011 (Horst) at 7. Horst teaches that the I/O Bus interface (4) contains "mapping and validation hardware translating th[e] virtual TNet address to the appropriate physical address." Ex. 1011 (Horst) at 5. The bus interface (4) generates the appropriate bus command on the bus with the proper physical address.

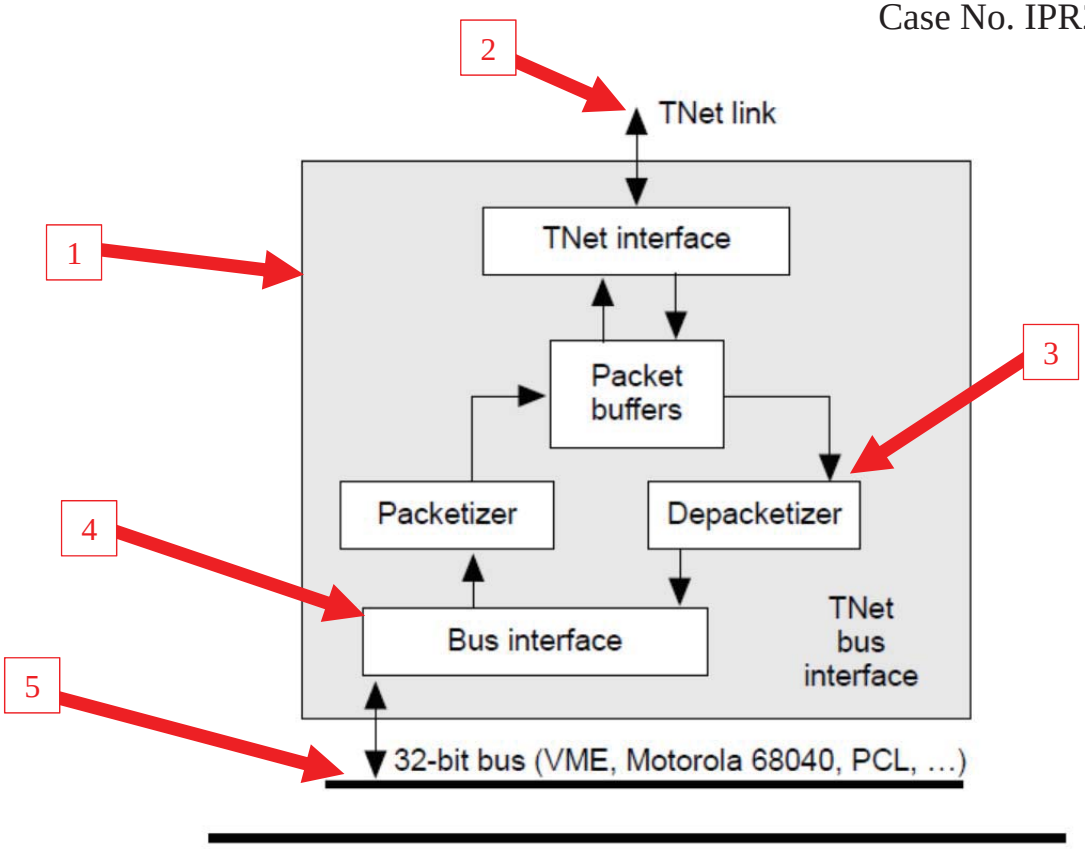


Figure 8. TNet bus interface.

Ex. 1011 (Horst) at 7, Fig. 8. For transaction occurring on the bus (5), Horst teaches that the TNet bus interface (1) discards the bus commands and address and generates TNet read-or-write transactions that travel through the TNet links (1). Ex. 1011 (Horst) at 7.

112. Horst teaches that, rather than using addresses to target processors or I/O over the TNet, TNet uses destination node IDs to route packets to individual nodes, and assigns each node an arbitrary 32-bit address space, rather than using flat addressing protocol, where every device is mapped into the same address space, to address components like PCI. Ex. 1011 (Horst) at 4-5, Fig. 5. In fact, Horst teaches that the assignment of a unique 32-bit address to each node is

contrary to the addressing scheme of most I/O busses, including PCI, which "[t]ypically, use one-to-one mapping" (e.g. are mapped to locations in the host CPU's physical address space rather than being assigned their own 32-bit address space). Ex. 1011 (Horst) at 5. TNet's addressing and node identification scheme means that if any given TNet transaction is routed to a different node than intended, it will cause the node to generate a completely different operation or transaction. For example, if a TNet packet is routed to a processor TNet node, the TNet processor node might generate a transaction with processor memory. If, on the other hand, the same TNet packet, with the same address, is routed to a SCSI node, the TNet to SCSI controller ASIC will generate a SCSI transaction. All this functionality requires significant a priori system-wide initialization of all TNet interfaces, including their address translation and access validation tables.

113. Further, Horst teaches that TNet transactions do not use PCI standard addresses, or any standard addressing scheme. First, TNet states that it uses virtual addresses and assigns each node its own virtual 32-bit address space. Ex. 1011 (Horst) at 2, 4-5. PCI standard addresses are based on three flat physical address spaces used by all memory and devices on the PCI bus. Ex. 2001 (PCI Spec Rev. 2.1) at 42 (memory, I/O, and configuration). As a result of its use of virtual addresses, no TNet transaction can contain a PCI standard address. Second, Horst states that TNet does not support memory-mapped I/O because Horst thought it

was the wrong model for the problem he was solving—interconnecting multiple processor nodes each with their own memory address space. Ex. 1011 (Horst) at 6. The PCI standard memory read-and-write transactions are memory-mapped I/O transactions, where the PCI devices are mapped into the host CPU's physical address space. Ex. 2014 at 13 (PCI Express Sys. Architecture at 122). The PCI standard encourages the use of memory-mapped I/O. Ex. 2001 (PCI Spec Rev. 2.1) at 42.

114. While there is a lack of detail around the exact routing mechanisms and address schemes used on a TNet link, it is clear from the TNet reference that there is no PCI standard bus transaction on the TNet link for another reason. The disclosed TNet transaction types are only read, write, and unacknowledged write. Ex. 1011 (Horst) at 5, Fig. 5. The TNet packets and transaction types do not allow for the PCI standard bus transactions (e.g., memory read and write, I/O read and write, configuration read and write). Ex. 1011 (Horst) at 4-5, Fig. 5; Ex. 2001 (PCI Spec Rev. 2.1) at 37. Nor does the Horst paper disclose any mechanism for the TNet packets to communicate PCI bus commands indicating a type of PCI standard bus transaction. Ex. 1011 (Horst) at 4-5; Ex. 2001 (PCI Spec Rev. 2.1) at 37. Instead, the only commands Horst discloses for TNet relate to link-level flow control, initialization, and error signaling. Ex. 1011 (Horst) at 3, Table 1.

115. Further, TNet transactions are not encoded PCI bus transactions. As the Board noted, the encoding disclosed in Horst is 8b/9b encoding. Ex. 1011 (Horst) at 3, Table 1. 8b/9b is a serial line code that uses a simple, reversible algorithm to represent 8-bit words in 9 characters so that the number of high and low signals on a serial channel can be balanced for clock regeneration. Ex. 1011 (Horst) at 3 (3 of 6 code). This 8b/9b line encoding in Horst is incapable of somehow encoding a PCI bus transaction into a TNet transaction. In fact, as Horst states, the 8b/9b encoding is applied to TNet transactions, not PCI transactions. Ex. 1011 (Horst) at 3.

116. Since TNet was addressed to a different problem than the claimed invention—interconnecting multiple processor nodes—the system was never universally adopted as a local I/O interconnect standard. One of the largest barriers to TNet's adoption as a local I/O interconnect standard is its proprietary nature requiring new drivers and ASICs at each processor and I/O node before it can be used with existing I/O devices and drivers. Ex. 2012 at 20 (CA at 531). If TNet had used PCI standard transactions, it might have been a viable alternative as a local bus protocol, but it would not have worked for its stated purpose as a processor node interconnection network. On the other hand, PCI Express, a PCI-specific system that transmits serialized PCI standard bus transactions, does not require the proprietary drivers and other issues involved in TNet. PCI Express has

become an industry standard as a local I/O interconnect and is ubiquitous today. However, PCI Express is not commonly used as a memory-fabric interconnect network protocol like Infiniband, TNet's successor.

D. Summary

117. In my opinion, neither Horst nor the secondary references disclose or render obvious the "encoded serial bit stream of Peripheral Component Interconnect (PCI) bus transaction" in claim 54 or "encoded address and data bits of PCI bus transaction" in claim 61 of the '873 patent. Horst was directed to a different problem—interconnecting multiple processor nodes, and therefore used a different solution.

V. CLAIM CONSTRUCTIONS

A. Peripheral Component Interconnect (PCI) bus transaction

118. In my opinion, the proper construction for PCI bus transaction is the one the Board adopted on institution—PCI standard bus transaction. In the preliminary proceedings to this case, Petitioner argued for an unreasonably broad construction of "Peripheral Component Interconnect (PCI) bus transaction" and "PCI bus transaction" as just a "bus transaction." At institution, this Board construed the terms to mean "Peripheral Component Interconnect (PCI) industry standard bus transaction." I agree with the Board's construction. Any other reading of the term "Peripheral Component Interconnect (PCI)" would render it

meaningless. The Board's construction is consistent with the understanding of the POSA in light of the specification.

119. A POSA would understand that a PCI bus transaction means just that—a PCI bus transaction. The PCI standard was well known at the time of the invention, and the POSA would not confuse the PCI standard with any other peripheral component bus. This was highlighted during the deposition of Mr. Young, EMC's expert. When asked the construction used for his analysis, he talked at length about how the PCI bus transaction required the PCI standard and that the POSA would understand that. Ex. 2022 (Young 5/20 Depo.) at 12:17-14:13. In fact, Mr. Young stated that the word "transaction" in the terms of the claims could not be taken out of context of the PCI bus transaction as a whole. Ex. 2022 (Young 5/20 Depo.) at 12:19-22; 13:9-17. Based on that, he stated that he had applied "one address phase followed by one more data phases on a PCI bus" in his declaration. Ex. 2022 (Young 5/20 Depo.) at 14:9-13. I note that a PCI bus is parallel and the claims discuss communicating PCI bus transactions serially, so the claims do not require that the PCI bus transaction occur on a PCI bus; they require the address and data phases of a PCI bus transaction, which include the PCI address and bus command information during the address phase (Ex. 2001 (PCI Spec Rev. 2.1) at 32), and the PCI data and byte enables during the data phases (Ex. 2001 (PCI Spec Rev. 2.1) at 32). Mr. Young stated essentially this when

asked what construction he applied to "an encoded serial bit stream of PCI bus transaction" describing it as "the information required to describe the PCI bus transaction mapped" "to some number of serial lines." Ex. 2022 (Young 5/20 Depo.) at 15:12-16:17. Mr. Young stated that his construction of PCI address bits was "the bits that represent the logical address of the PCI bus transaction ordered in such a way to be sent down one or more serial lines." Ex. 2022 (Young 5/20 Depo.) at 22:11-16. In other words, the PCI standard address in serial form. It wasn't until later that Mr. Young changed his position and remembered that he had used a much broader definition for his analysis that did not involve PCI at all and conflicted with the Board's construction. Ex. 2022 (Young 5/20 Depo.) at 23:24-24:14; 24:25-25:4; 26:18-27:2.

120. Therefore, the Board's construction of "PCI bus transaction" as "PCI standard bus transaction" is the proper construction.

B. Encoded

121. Several claims utilize the word encoded in some fashion. Encoding can apply to different types operations in the field of computing; however, two conditions are required for all encoding operations involving numerical values. First, as Mr. Young states, encoding must be reversible—that is a recipient must be able to accurately decode the information. Ex. 2022 (Young 5/20 Depo.) at 8:23-25; 10:12-11:18. Second, coded values must uniquely relate to unencoded values;

in other words, every coded value must relate to one and only one unencoded value. Ex. 2024 at 4 (1994 IBM computer dictionary at 111)(code (1) "A set of rules that maps the elements of one set onto the elements of another set. … The first set is the coded set and the second set is the code element set. An element of the code element set may be related to more than one element of the coded set *but the reverse is not true*."). While encoding can typically have a variety of meanings within this framework, the '873 patent describes a very limited set of operations that could be considered encoding. In fact, Petitioner's expert, Mr. Young, opined that the '873 patent only described one form of what a POSA would consider encoding—converting parallel information into serial information, the ordering of the serial bits, and which serial line the bit is directed to. Ex. 2022 (Young 5/20 Depo.) at 9:1-22; 10:1-7. I agree; these are generally the only forms of encoding discussed in the '873 patent. In the language of the '873 patent, these are discussed as turning PCI bus signals into bits for serial transmission, placing the PCI bus transaction bits in 10-bit packets, and placing the PCI bits on multiple serial channels.

122. Encoding PCI bus signals into bits is described in the specification as encoding the control signals into control bits. Ex. 1001 ('873 patent) at 5:33-36. A POSA would recognize that this is a form of encoding. The specification goes on to explain that "the control bits representing control signals are decoded back

into PCI control signals" before being transmitted on the PCI bus. Ex. 1001 ('873 patent) at 5:36-39.

123. Encoding PCI bus transactions into 10-bit packets is described in the '873 patent specification. Ex. 1001 ('873 patent) at 17:54-60 (A/D bits and some control bits serialized into 10-bit packets); 21:65-67 (XPBus is cable friendly and transmits fixed length data packets within a clock cycle). The '873 patent states that serial bit channels "typically transmit bit packets and use a serial data packet protocol." Ex. 1001 ('873 patent) at 21:61-65. A POSA would recognize packetizing serial bits as a form of encoding.

124. The ordering form of encoding is described in the specification in the description of the encoders that "format the PCI address/data bits to a form more suitable for parallel to serial conversion" before being sent over the serial line. Ex. 1001 ('873 patent) at 16:55-58; 17:11-13. The ordering form of encoding is also shown through the multi-channel LVDS layout in Figures 13-15. Ex. 1001 ('873 patent) at FIGS. 13-15. The decoders on the receiving end of the line are then able to reverse the ordering for transmission over the PCI bus. *Id.* at 16:58-60.

125. Therefore, the BRI of "encode" in light of the specification is "a reversible operation that turns a signals into bits, packetizes bits into a specified size packet, or orders bits onto one or more serial transmission lines."

VI. THE CHALLENGED CLAIMS ARE PATENTABLE OVER THE INSTITUTED GROUNDS

126. It is Petitioner's burden to show by a preponderance of the evidence that the challenged claims are unpatentable. 35 U.S.C. § 316(e). Petitioner has failed to meets its burden of proof for any of the alleged grounds of unpatentability that it has asserted, i.e. 35 U.S.C. §§ 102 and 103.

B. The Challenged Claims Are Patentable Under 35 U.S.C. § 103

127. EMC's arguments as to why Horst and the secondary references invalidate the challenged claims fail to teach a LVDS serial channel that transmits an encoded serial bit stream of a PCI bus transaction. Horst was designed for the purpose of interconnecting multiple separate computer systems for sharing processor and memory resources, while the claimed invention and the PCI standard dealt with interconnecting a processor node and its local I/O devices. The PCI standard would not work to interconnect multiple processor nodes. As a result, the Horst reference does not transmit serial PCI bus transactions and it would not be obvious to modify Horst to add this limitation.

C. EMC's Expert Mr. Young Did Not Form an Opinion that Horst Transmitted Serial PCI Bus Transactions

128. Mr. Young, EMC's expert, did not form an opinion in his declaration (Ex. 1003) that any of the grounds disclosed or rendered obvious communicating address and data bits of a PCI bus transaction in serial form. Although I note

Mr. Young initially stated during his deposition that the ordinary meaning of PCI bus transaction was a transaction in accordance with the PCI local bus standard, consistent with the Board's construction (Ex. 2022 (Young 5/20 Depo.) at 15:12-16:17), this is not the construction Mr. Young applied in forming his invalidity opinions in his declaration. Ex. 2022 (Young 5/20 Depo.) at 36:1-9. Instead, Mr. Young stated that he used "peripheral bus transaction" in forming his opinions on invalidity, reading "PCI standard" out of the claims. Ex. 2022 (Young 5/20 Depo.) at 36:1-9. By reading "PCI standard" out of the claim, Mr. Young was able to assert that a TNet transaction or a SCI transaction was a "bus transaction." But TNet transactions are not PCI standard transactions in serial form.

129. Rather, TNet transactions have their own protocol and conform to their own design guidelines and standards and are used for different applications than the PCI local bus standard. TNet is used as a high-speed interconnect for multiple processor nodes. In fact I used Infiniband interconnections (a successor of TNet) to share processing power in building the L-CSC supercomputer. Intelligent processors and drivers are required to translate between TNet and PCI standard bus transactions.

130. In my opinion, the fact that the claims require a PCI standard bus transaction is a key component of the claimed invention's value and utility for their intended purpose. The PCI standard was immensely popular at the time of the

'873 patent's invention and maintenance of the PCI standard allowed interoperability between the ACM and a multitude of PCI standard drivers and peripheral components. Ex. 1001 ('873 patent) at 4:50-58. Without maintaining the PCI standard, the ACM would require custom drivers and/or peripheral components in order to function, significantly reducing the likelihood the ACM would be a viable product in the consumer marketplace.

131. Mr. Young's application of a construction of "PCI bus transaction" as simply a generic "peripheral bus transaction," (Ex. 2022 (Young 5/20 Depo.) at 30:8-15; 32:4-19) broadened the construction of PCI transaction by specifically excluding the "PCI" limitation. Ex. 2022 (Young 5/20 Depo.) at 30:16-24; 32:13-19. Mr. Young went on to state that he did not use the Board's construction of a "PCI bus transaction" as a "PCI standard bus transaction" in forming his validity opinions and, instead, simply applied the broader construction of "peripheral bus transaction." Ex. 2022 (Young 5/20 Depo.) at 31:13-32:3; 32:13-19; 36:1-9. I have excised some relevant portions of Mr. Young's deposition transcript and included them below:

> Q: … for all the construction you used when you deal with PCIbus [stet] transaction, your construction was broader than the PCI standard?
>
> A: Yes. (Ex. 2022 (Young 5/20 Depo.) at 36:1-9.)
>
> * * *
>
> Q: You are not limiting your construction to the PCI standard?

> A: I'm not limiting it to PCI standard. (Ex. 2022 (Young 5/20 Depo.) at 32:17-19.)
>
> * * *
>
> Q: And in your declaration you gave no pinions regarding the validity based on the construction [of PCI bus transaction] eventually adopted by the Board?
>
> A: That construction had not been proposed by the Board at that point in time. So I couldn't use that the time.
>
> Q: … In your declaration you gave no opinions about any alternative constructions of data bits of PIC bus transaction?
>
> A: I don't believe so. (Ex. 2022 (Young 5/20 Depo.) at 31:17-32:3.)

132. As a result of Mr. Young's disregarding the "Peripheral Component Interconnect (PCI)" term in forming his opinions, Mr. Young's opinions with regard to every claim limitation that requires a "PCI bus transaction" are unreliable and unsupportable under the proper reading of "PCI bus transaction." These claim limitations include:

i. Claim 54 -- "encoded serial bit stream of Peripheral Component Interconnect (PCI) bus transaction"

ii. Claim 61 – "encoded serial bit stream of PCI bus transaction comprises encoded PCI address and data bits"

1. Claims 54 and 56-61 are not rendered obvious over Horst, and the LVDS Owner's Manual; (Ground 1) and further in view of either Pocrass (Ground 2) or Deters (Ground 3)

133. Horst does not sufficiently disclose the claimed limitations of claims 54 and 61. Specifically, Horst does not disclose LVDS serial channels that

transmit an encoded serial bit stream of Peripheral Component Interconnect (PCI) bus transactions as required by claim 54, or encoded PCI address and data bits as required by claim 61. Further, to modify Horst to transmit PCI bus transactions would render the system inoperable.

a. Claim 54 and 56-61 – LVDS channels for communicating "encoded serial bit stream of Peripheral Component Interconnect (PCI) bus transaction"

134. There is no evidence that the north bridge disclosed in TNet transmits an "encoded serial bit stream of PCI bus transaction" over the TNet network. Instead, Horst suggests that the only things transmitted over the TNet network are encoded TNet packets.

135. I understand the Board found that, because the TNet serial network connects to a PCI bus through a TNet to PCI bus interface, this was sufficient to institute a review to determine if the TNet serial network must communicate an encoded serial bit stream of a PCI bus transaction. It does not.

136. Just because the TNet bus interface causes a PCI transaction on a PCI bus does not mean that a PCI standard transaction was transmitted across the TNet link. Instead, a TNet transaction communicates over the TNet to the TNet bus interface. The TNet bus interface then uses its knowledge of the PCI bus and the

PCI bus protocol (something the TNet transaction has no knowledge of) to generate a PCI transaction on the PCI bus.

137. Claim 54 requires an "encoded serial bit stream of Peripheral Component Interconnect (PCI) bus transaction" and claim 61 additionally requires the encoded serial bit stream of PCI bus transaction include "encoded PCI address and data bits." This requires that PCI standard bus commands, address, and data bits be communicated over the TNet links. Paper No. 14 (Institution Decision) at 5-7. The PCI standard sets out specific requirements for PCI bus commands and standard addresses. Critically, all PCI addresses are physical addresses in one of three memory spaces. Ex. 2001 (PCI Spec Rev. 2.1) at 42. The only addresses disclosed in Horst that are communicated over TNet are virtual TNet addresses. Ex. 1011 (Horst) at 5, Fig. 5.

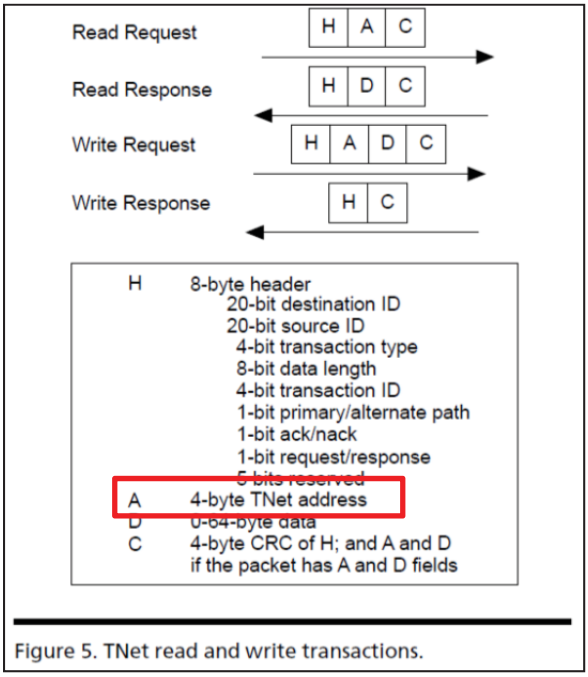


Figure 5. TNet read and write transactions.

138. Horst states that TNet addresses are virtual addresses, not physical addresses. Ex. 1011 (Horst) at 2, 6. This means that TNet addresses cannot be PCI standard addresses, which are physical addresses. This makes sense because PCI standard addresses using a flat addressing scheme (a single processor node address space) would not work in TNet because TNet interconnects multiple processor nodes, each with its own address space.

139. In addition, TNet addresses cannot be encoded PCI standard addresses. First, the BRI of encoded in the '873 patent is limited to converting signals to bits, forming bits into packets, or ordering bits onto multiple serial channels. Translating a physical PCI address to a virtual TNet address does not fall within any of the three types of encoding disclosed in the specification.

Second, as Mr. Young stated, encoding must be reversible. Ex. 2022 (Young 5/20 Depo.) at 8:23-25; 10:12-11:5. As discussed above, virtual address translation is not reversible and, therefore TNet virtual addresses cannot be encoded PCI addresses under the BRI or EMC's definition. Third, encoding mathematical processes, like binary address data, must maintain a one-to-one relationship between the encoded set and the unencoded set. Ex. 2024 at 4 (IBM Tech Dictionary at 111). For BRI, the reason for requiring a one-to-one relationship between encoded data and unencoded data is so that decoding will result in one ascertainable value. If an encoded value mapped to multiple different unencoded values, there is no hint which unencoded value is correct. Horst discloses that the virtual TNet addresses relate to multiple physical addresses, because each TNet node has its own 32-bit address space. For this additional reason, virtual TNet addresses cannot be PCI standard addresses.

140. Further, Horst affirmatively teaches a POSA that serialized PCI transactions are not transmitted over TNet. This makes sense because PCI transactions would not work in the multiple processor node environment that TNet was designed for. First, the PCI standard uses three physical address spaces, memory address space, I/O address space, and configuration address space. Ex. 2001 (PCI Spec Rev. 2.1) at 42. The PCI address space used for a particular transaction is dictated by the PCI bus command for the transaction. Ex. 2001 (PCI

Spec Rev. 2.1) at 25, 42. Horst discloses that TNet does not use any of these PCI standard transaction types. Ex. 1011 (Horst) at 4-5, Fig. 5 (only read, write, and unacknowledged write).

141. Nor is the inclusion of PCI bus commands or transaction types inherent to TNet. The TNet PCI interface uses an intelligent ASIC, which a POSA would understand generates PCI bus commands based on the particular PCI bus configuration. As a result, the TNet transaction need not be a serialized PCI standard bus transaction for the intelligent TNet PCI interface ASIC to generate a PCI bus transaction on the PCI bus. This makes sense because TNet needs to interact with multiple processors and bus interfaces, each connecting to a different bus protocol with different devices. Designing a TNet protocol that transmitted every different bus protocol for each bus configuration would be extremely complicated and would defeat Horst's stated goal of having a simple transaction set. Ex. 1011 (Horst) at 4-5, Fig. 5.

142. Second, Horst teaches a POSA that PCI memory transactions are not supported. PCI memory transactions are memory mapped. Ex. 2014 at 13 (PCI Express Sys. Architecture at 122); Ex. 2015 at 10, 12 (PCI/PCI-X book at 83, 1143). This means that a PCI device is mapped to a particular range in the PCI bus host processor's memory address space. The PCI specification states that PCI memory transactions are the preferred transaction type. Ex. 2001 (PCI Spec Rev.

2.1) at 42. Horst states that TNet does not support memory-mapped I/O because the TNet designers thought it was the wrong model for their system. Ex. 1011 (Horst) at 6. As a result, Horst teaches a POSA that TNet does not transmit PCI standard memory transactions.

143. Third, the PCI standard is not capable of configuring any devices across the TNet link because there are multiple separate address spaces. As a result, the TNet PCI interface must be non-transparent to the PCI controller. See Ex. 2017 at 3 (VITA Journal at 6-7). That the TNet/PCI interface is non-transparent means the PCI controller is unaware of any devices on the other side of the TNet/PCI interface and therefore does not attempt to configure them. As a result, a POSA would understand that no PCI configuration transactions are sent over the TNet. This makes sense because the PCI standard is designed for interconnecting the components of a single computer system using a single address space, while TNet is designed to network multiple processor nodes, each with individual address spaces.

144. The fact that the TNet bus interfaces must be non-transparent also means that the TNet processor interface does not know the PCI address of the PCI component on the remote PCI bus. Accordingly, the TNet processor interface does not communicate an encoded serial bit stream of PCI bus transaction because at least the address bits are not included, which are required for a PCI bus transaction.

Instead, the TNet processor interface knows the node address for the TNet bus interface. Upon receipt of the TNet packet, the TNet bus interface can look up the PCI address of the PCI component in its memory, or the bus master's memory, using its address translation logic. Ex. 1011 (Horst) at 6. Of course, there are multiple details that can be modified in this process, but the point is that TNet does not disclose the transmission of PCI standard bus transactions or PCI standard address bits. Further, TNet does not need to transmit PCI standard bus transactions or address bits to allow the TNet bus interface to generate a PCI bus transaction.

145. Further, the fact that TNet cannot transmit PCI configuration transactions teaches a POSA that TNet cannot transmit PCI transactions because there is no way to configure the TNet nodes to work with PCI transactions. PCI configuration transactions are required by the PCI specification. Ex. 2015 at 10 (PCI/PCI-X book at 83). PCI configuration transactions are used to initialize the PCI bus and map the PCI devices addresses. Without PCI configuration transactions, the TNet nodes will not be configured to use PCI transactions.

146. While the disclosure in Horst is not detailed enough to explain the details of how the TNet system operates, the disclosure in Horst outlined above is sufficient to teach a POSA that PCI standard bus transactions and address are not transmitted over TNet. A POSA would expect this to be the case because TNet serves a different purpose than the PCI standard.

b. It would not be obvious to modify TNet to communicate PCI bus transactions over the TNet network

147. It is not obvious to merely replace a TNet transaction with a PCI bus transaction. The Board found a reasonable likelihood that Horst disclosed communicated address and data bits of a PCI bus transaction in serial form over TNet. The Board did not rely on potential obviousness. However, I provide an analysis of why it would not be obvious to use PCI standard bus transactions over TNet for illustration purposes.

(1) Horst discloses a functional TNet system, a POSA would have no reason to modify it to transmit PCI standard addresses or bus transactions over the TNet network.

148. Horst discloses a working TNet system that was specially designed to operate in the manner disclosed. Ex. 1011 (Horst) at 1. Horst further discloses that TNet is not designed to replace any existing LAN or I/O bus but, instead, is a new interconnection layer. Ex. 1011 (Horst) at 1. A POSA would have no reason to modify TNet to transmit PCI standard bus transactions instead of TNet transactions, particularly when TNet was custom designed for the purpose it serves and the PCI standard is designed for the different purpose of interconnecting a processing and memory node with its local I/O devices.

(2) Horst teaches away from modifying TNet to transmit PCI standard addresses or bus transactions over the TNet network

149. Horst teaches that, in designing TNet, they evaluated existing bus standards. Ex. 1011 (Horst) at 2. PCI was an existing bus standard at the time TNet was designed, as evidenced by the TNet PCI interface. Horst teaches that no existing standard met all the requirements for TNet so they "were forced [] into the difficult job of designing a completely new network." Ex. 1011 (Horst) at 2. As a result, Horst teaches that PCI was considered and rejected in favor of designing a new TNet system. Again, a POSA would not find this surprising because TNet is meant to interconnect multiple processor nodes with independent address spaces, something the PCI standard addressing scheme cannot do without specialized additional equipment.

150. Horst also teaches that memory-mapped I/O, which includes the preferred PCI standard bus transactions (Ex. 2001 (PCI Spec Rev. 2.1) at 42), are not supported because they thought it was the wrong model for TNet. Ex. 1011 (Horst) at 6. Again, this is not surprising since TNet has multiple processor nodes with multiple separate address spaces.

151. Rather than using PCI, Horst discloses creating an entirely new and proprietary network.

(3) Sending PCI bus transactions over the TNet network would render the TNet system disclosed in Horst inoperable

152. Using PCI standard bus transactions over the TNet network would render TNet inoperable. Again, this is not surprising since PCI is a local bus standard for interconnecting a single processor and memory node with its local I/O devices, whereas TNet is designed to interconnect multiple processor nodes, each with a separate memory address space. First, Horst discloses that TNet operates in a peer-to-peer fashion with multiple distributed intelligent nodes. Ex. 1011 (Horst) at 1-2. Horst also teaches TNet uses separate 32-bit virtual address spaces for each TNet node. The PCI standard, on the other hand, is a local bus protocol designed to interconnect a single processing and memory node to its local peripheral devices. In other words, the PCI bus standard is designed to interconnect components within the same box controlled by a single processing and memory unit.[1] The PCI standard relies on the processor's flat physical address spaces to address devices on the PCI bus. As a result, a PCI standard bus transactions flat addressing scheme is incapable of addressing the multiple TNet nodes, each with a separate 32-bit virtual address space.

[1] By single processing and memory unit, I mean either a single-core CPU, or multi-core CPU (such as an Intel Core i7 processor with 4 physical cores and 8 virtual cores) that is running a single operating system and has a single address space.

153. Second, the PCI physical addresses would not correspond to the same address in each TNet node because each TNet node relies on virtual addresses which are assigned arbitrarily. Therefore, PCI standard bus transactions physical addresses would result in address confusion with the TNet node virtual address spaces.

154. Third, each TNet processor and bus node is specifically designed to operate with TNet transactions, including TNet addresses and transactions. TNet processor and bus nodes would not function with PCI bus standard transactions requiring them to be redesigned or replaced.

(4) In order to allow PCI bus transactions over the TNet network, it would require extensive redesign and add unnecessary complexity and expense to a functioning system

155. In order to make TNet work with PCI bus transactions, each processor node would first need to be isolated from every other processor node using non-transparent bridges to separate the TNet processor nodes address spaces from each other. Ex. 2016 at 6 (Intel Nontransparent bridge spec at 2); Ex. 2017 at 3-4 (VITA Journal at 6-7). Second, each TNet I/O node would need to be assigned to a single processing unit and redesigned to operate under that processing unit's physical address space, rather than with its own 32-bit virtual address space in order to allow the PCI standard addressing scheme to work. Third, each TNet

interface would need to be redesigned to operate with PCI configuration transactions, and at least one of either PCI memory or I/O transactions, in order to be functional. These necessary redesigns to allow PCI transactions over TNet would at least require adding new hardware and writing new driver software. Modifying TNet use PCI transactions over TNet might also require additional hardware redesign.

(5) The combination of PCI with TNet is merely hindsight reasoning

156. The computer industry went with PCI and PCI Express for local bus and local I/O interconnections. While Horst explains that he thought memory-mapped I/O was "the wrong model" for a multiple processor node system with multiple address spaces, memory-mapped I/O was the model that succeeded inside a single computer system. Both views are correct in the appropriate context. PCI is a bus standard used inside a computer with a flat address model. TNet is a network that was invented to support a system of thousands of computer nodes with their own address spaces, which required virtual addresses and validation. At this point, combining PCI with a model that teaches away from the combination because of its different purpose is merely hindsight reasoning.

VII. CONCLUSION

157. I reserve the right to offer opinions relevant to the invalidity of the '873 patent claims at issue and/or offer testimony in support of my Declaration.

158. In signing this Declaration, I recognize that this Declaration will be filed as evidence in a contested case before the Patent Trial and Appeal Board of the United States Patent and Trademark Office. I also recognize that I may be subject to cross-examination in the case. If required, I will appear for cross-examination at the appropriate time.

I hereby declare that all statements made herein of my own knowledge are true, and that all statements made on information and belief are believed to be true, and, further, that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under 28 U.S.C. § 1001.

Dated: June 10, 2015

Respectfully submitted,

Volker Lindenstruth

Appendix A

Process flow of a processor writing to a display over a PCI local bus.

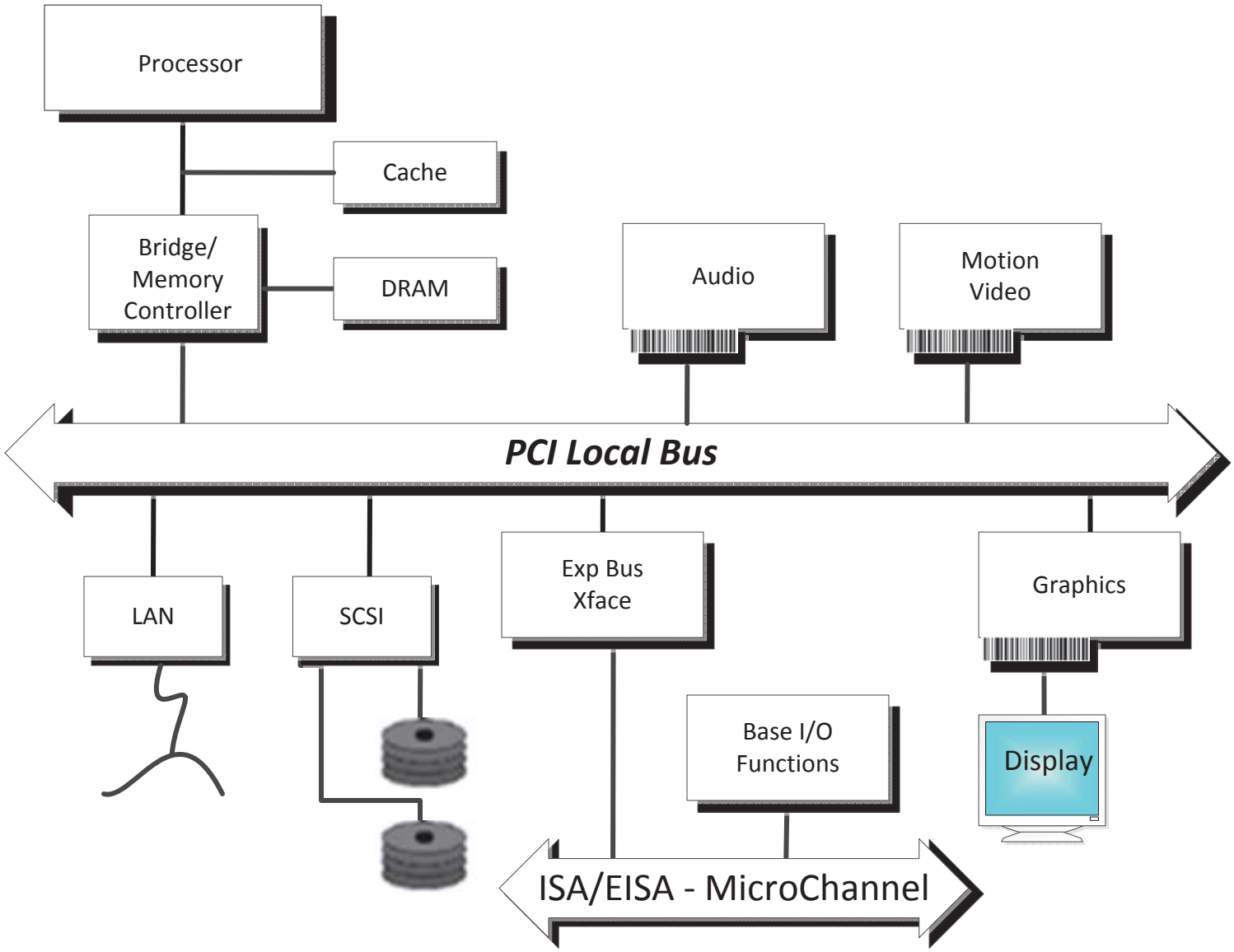


Figure 1-2: PCI System Block Diagram

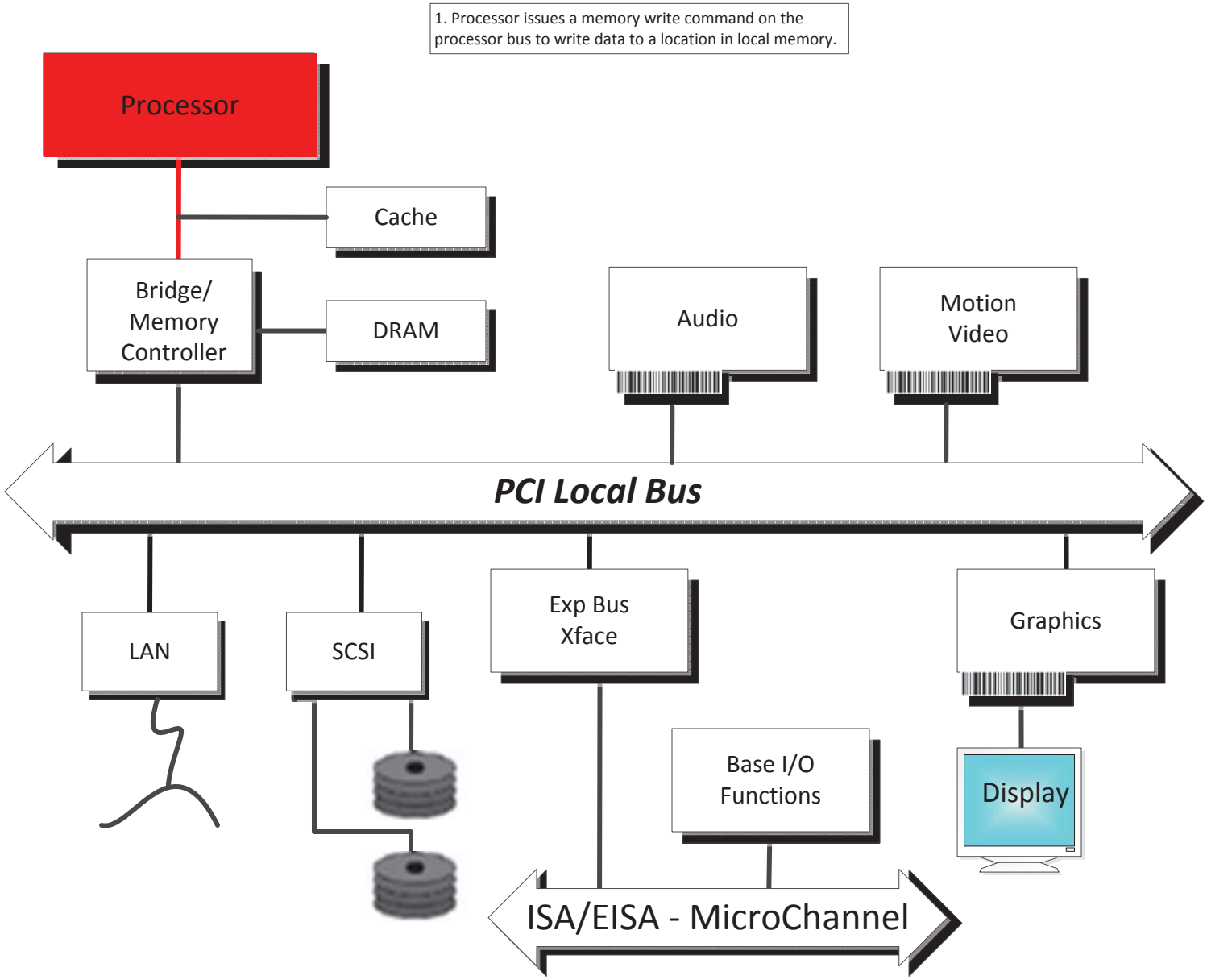
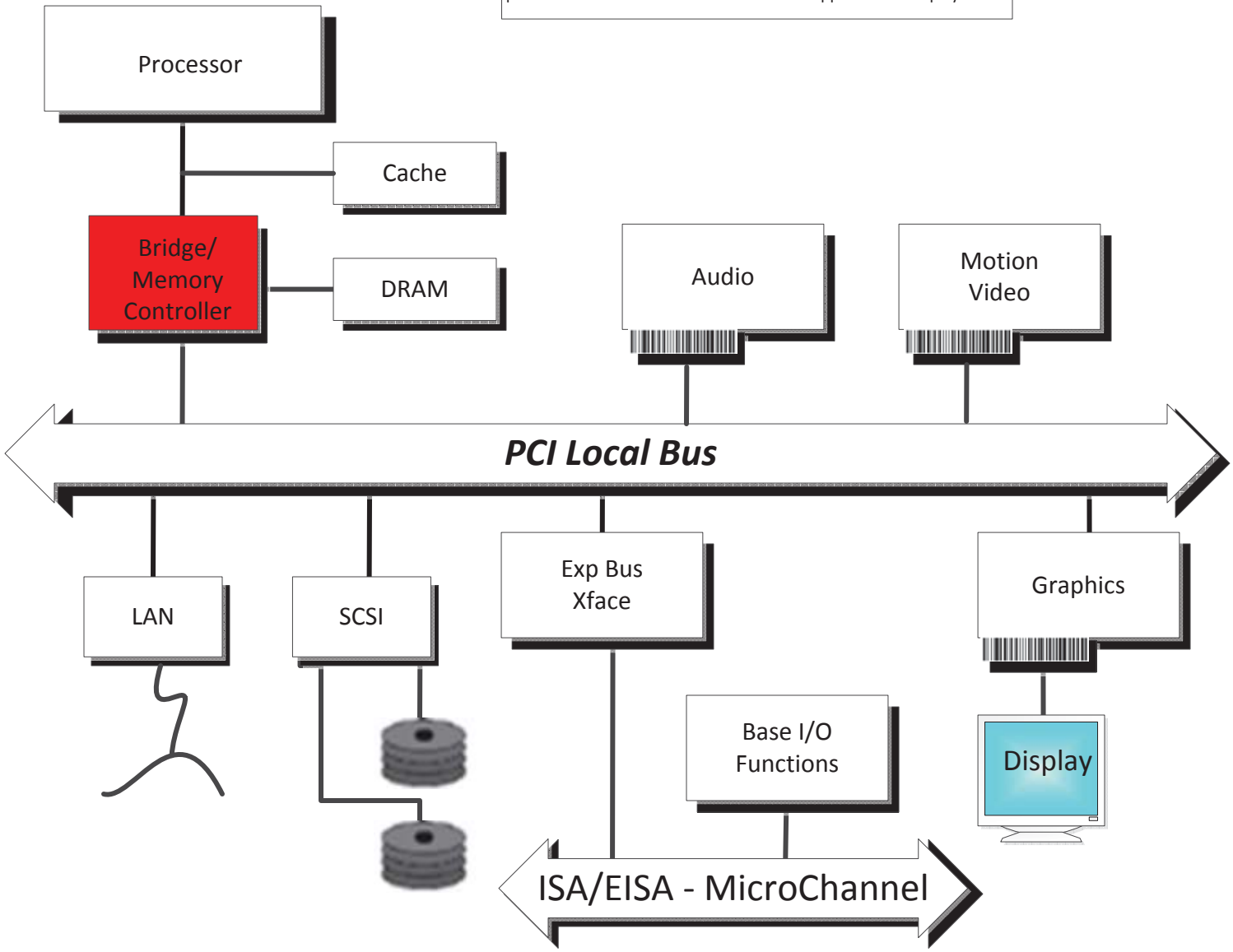


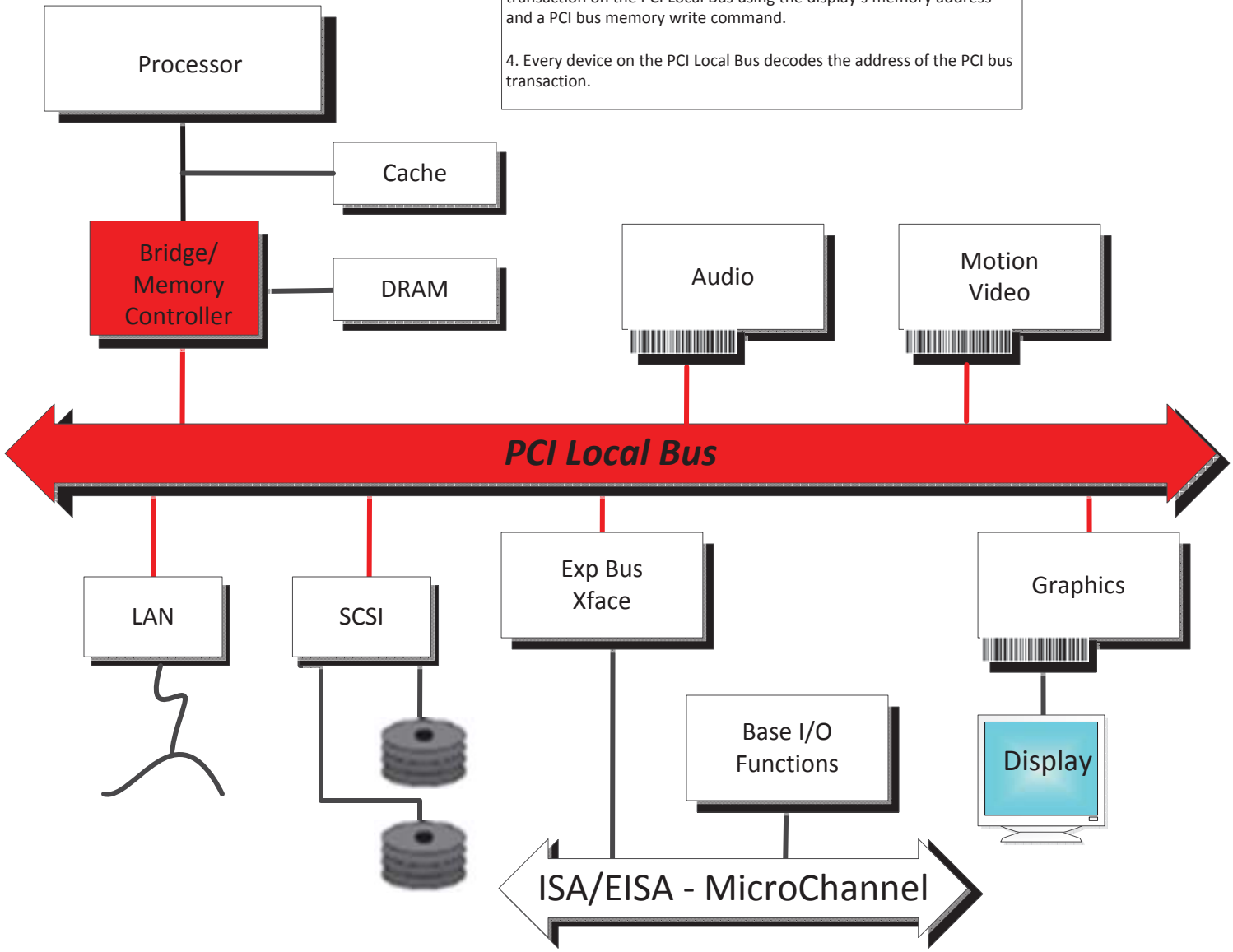
Figure 1-2: PCI System Block Diagram

2. The memory controller receives the write command from the processor and determines the address is mapped to the Display.



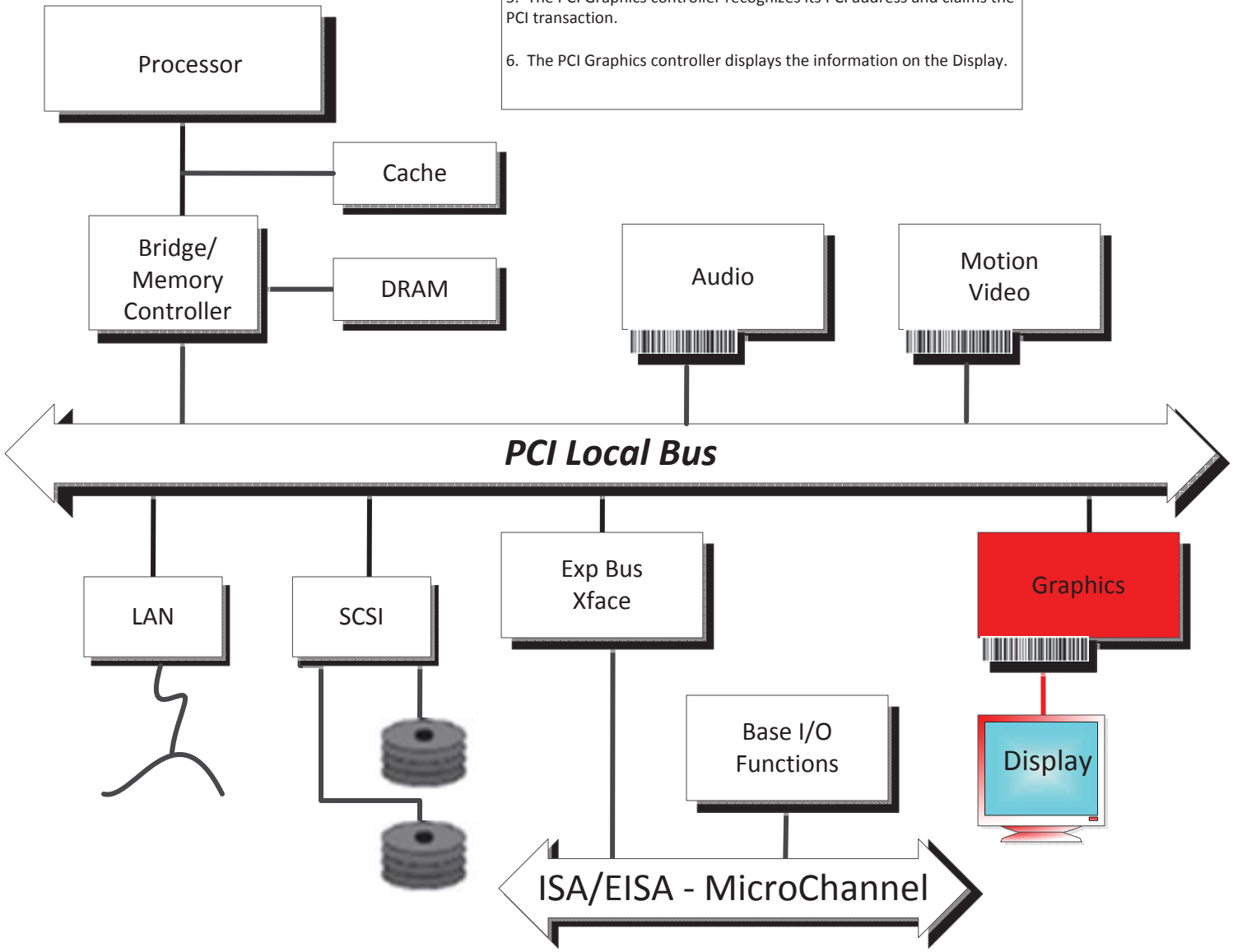
3. The memory controller generates a memory write PCI bus transaction on the PCI Local Bus using the display's memory address and a PCI bus memory write command.

4. Every device on the PCI Local Bus decodes the address of the PCI bus transaction.

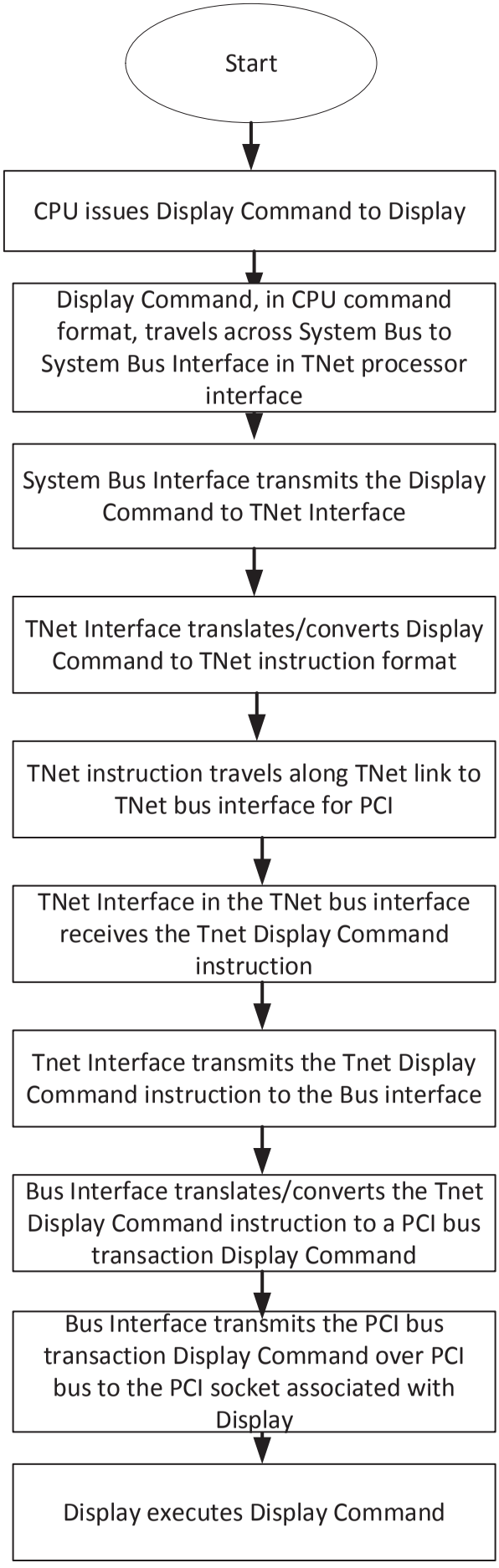


5. The PCI Graphics controller recognizes its PCI address and claims the PCI transaction.

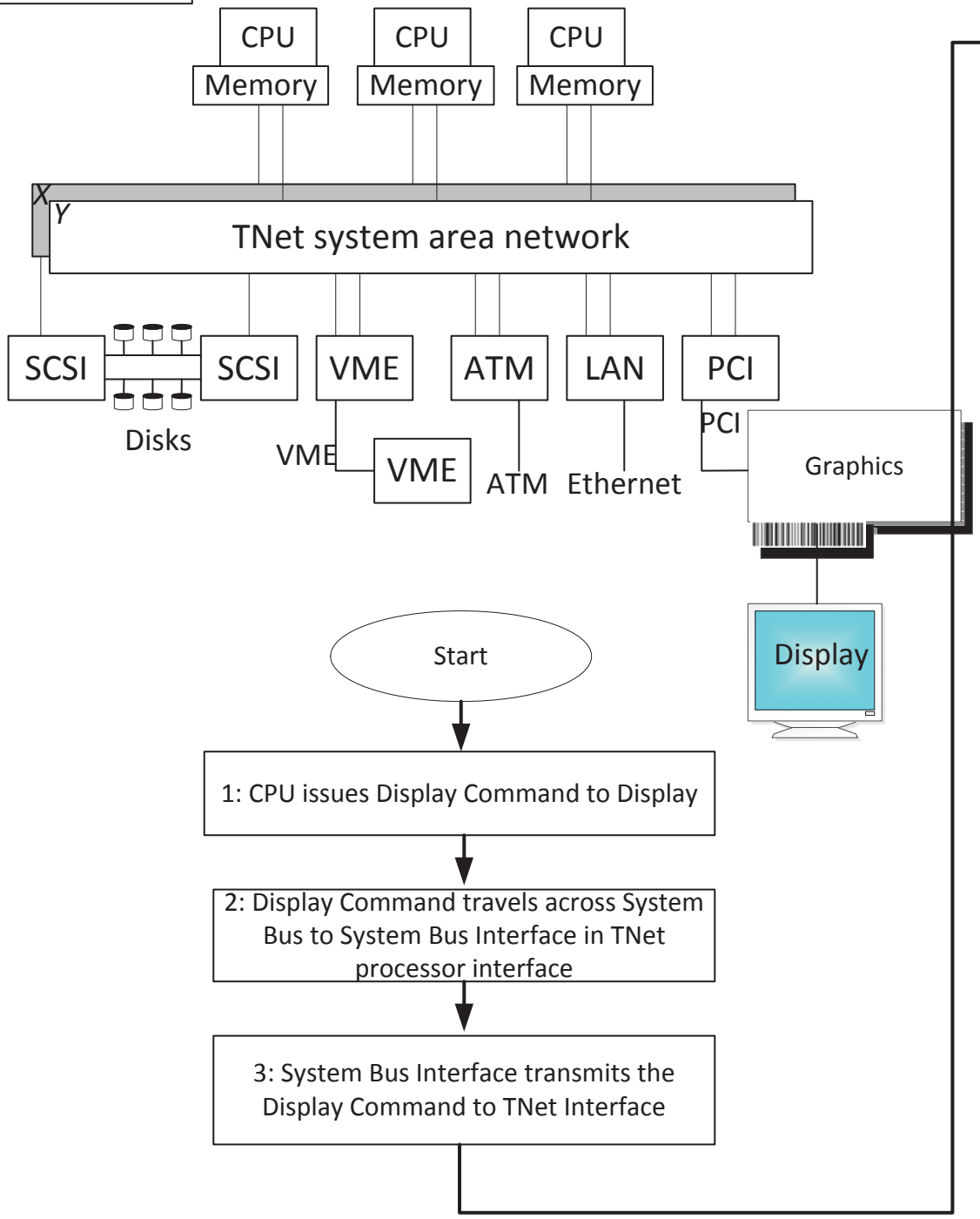
6. The PCI Graphics controller displays the information on the Display.



TNet Instruction Flow



TNet Instruction Flow



Start

1: CPU issues Display Command to Display

2: Display Command travels across System Bus to System Bus Interface in TNet processor interface

3: System Bus Interface transmits the Display Command to TNet Interface

4: TNet Interface translates/converts Display Command to TNet instruction format

5: TNet instruction travels along TNet link to TNet bus interface for PCI

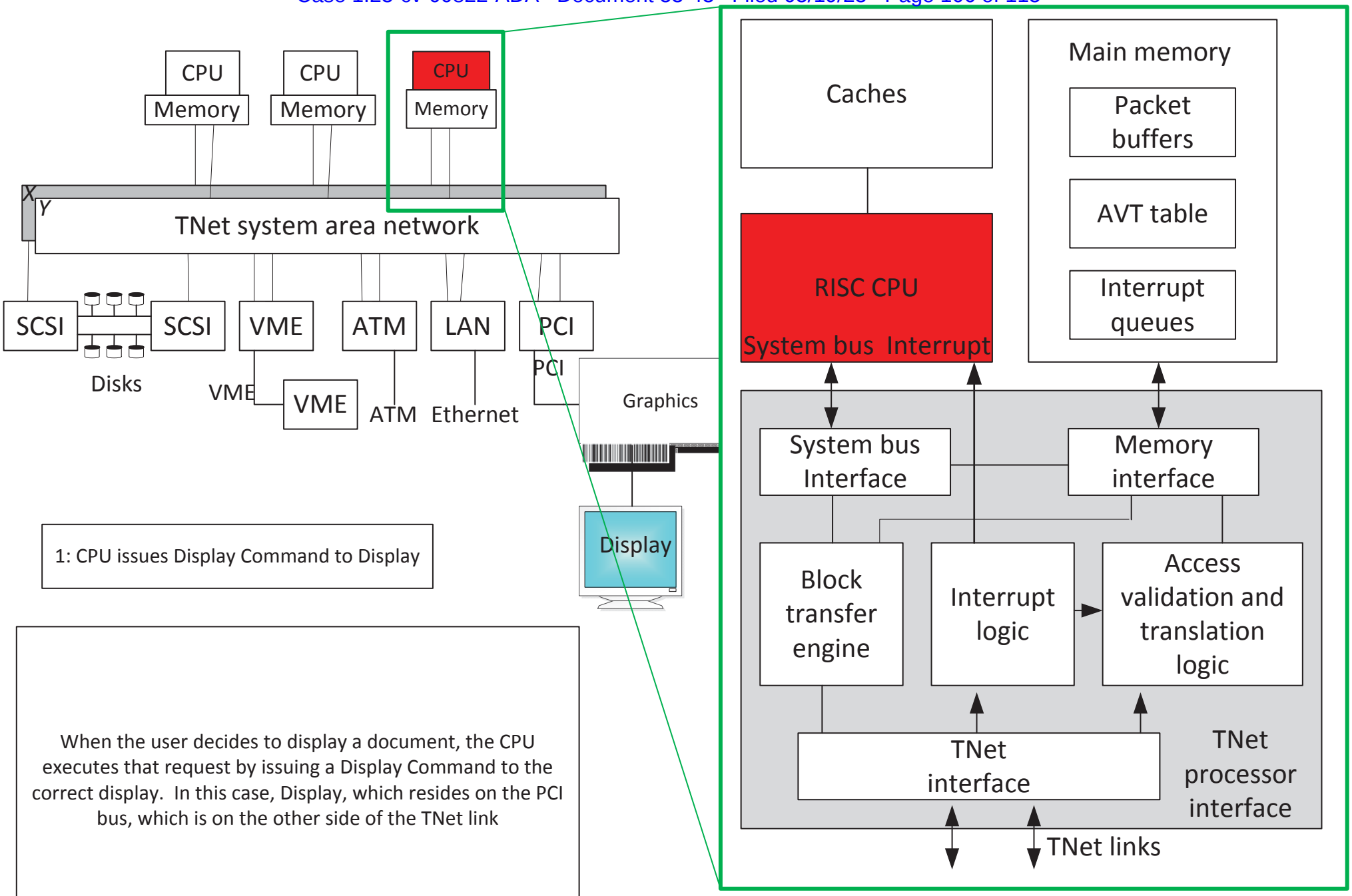
6: TNet Interface in the TNet bus interface receives the Tnet Display Command instruction

7: Tnet Interface transmits the Tnet Display Command instruction to the Bus interface

8: Bus Interface translates/converts the Tnet Display Command instruction to a PCI bus transaction Display Command

9: Bus Interface transmits the PCI bus transaction Display Command over PCI bus to the PCI socket associated with Display

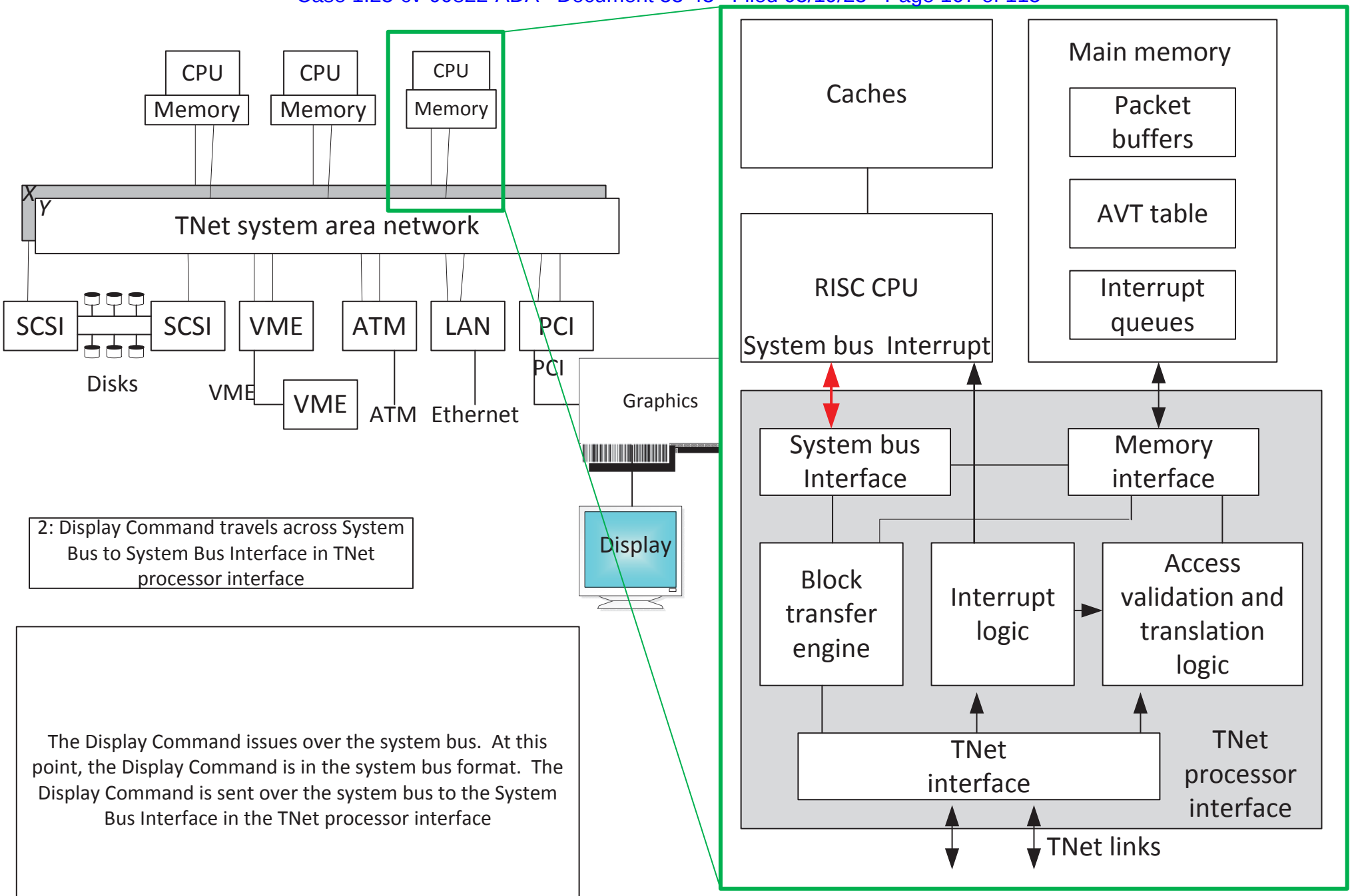
10: Display executes Display Command



1: CPU issues Display Command to Display

When the user decides to display a document, the CPU executes that request by issuing a Display Command to the correct display. In this case, Display, which resides on the PCI bus, which is on the other side of the TNet link

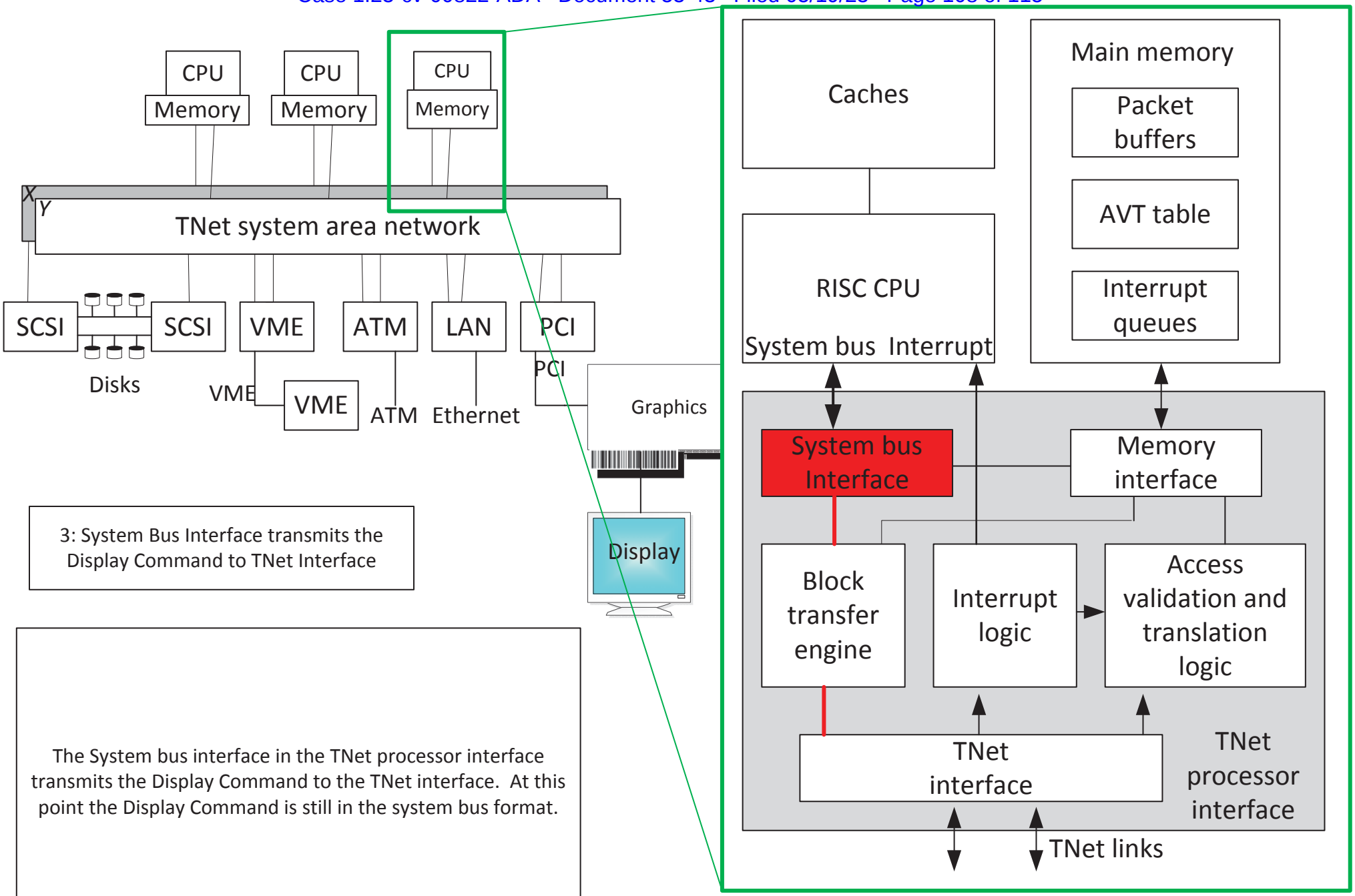
TNet Instruction Flow
Step 1



2: Display Command travels across System Bus to System Bus Interface in TNet processor interface

The Display Command issues over the system bus. At this point, the Display Command is in the system bus format. The Display Command is sent over the system bus to the System Bus Interface in the TNet processor interface

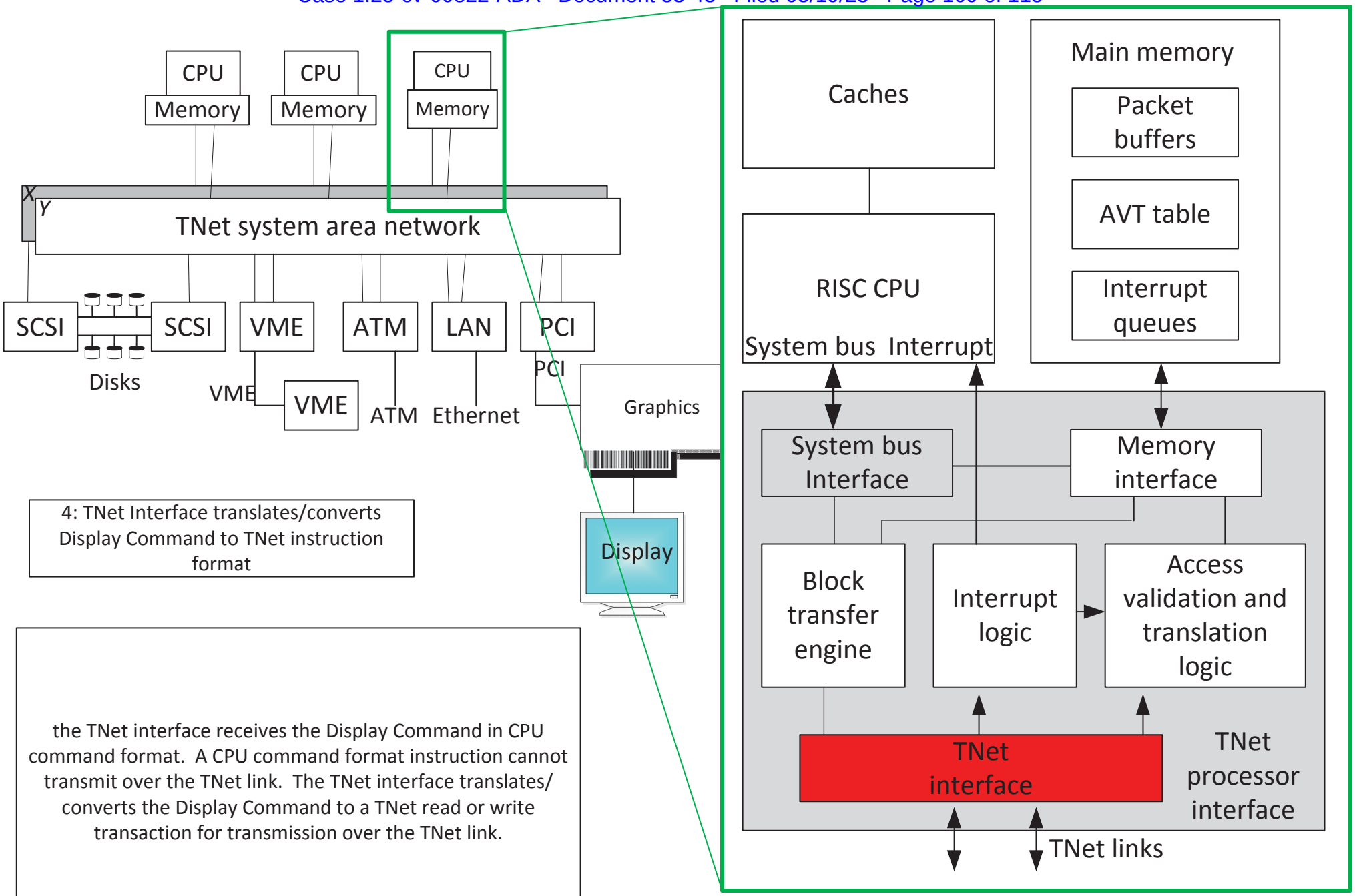
TNet Instruction Flow
Step 2



3: System Bus Interface transmits the Display Command to TNet Interface

The System bus interface in the TNet processor interface transmits the Display Command to the TNet interface. At this point the Display Command is still in the system bus format.

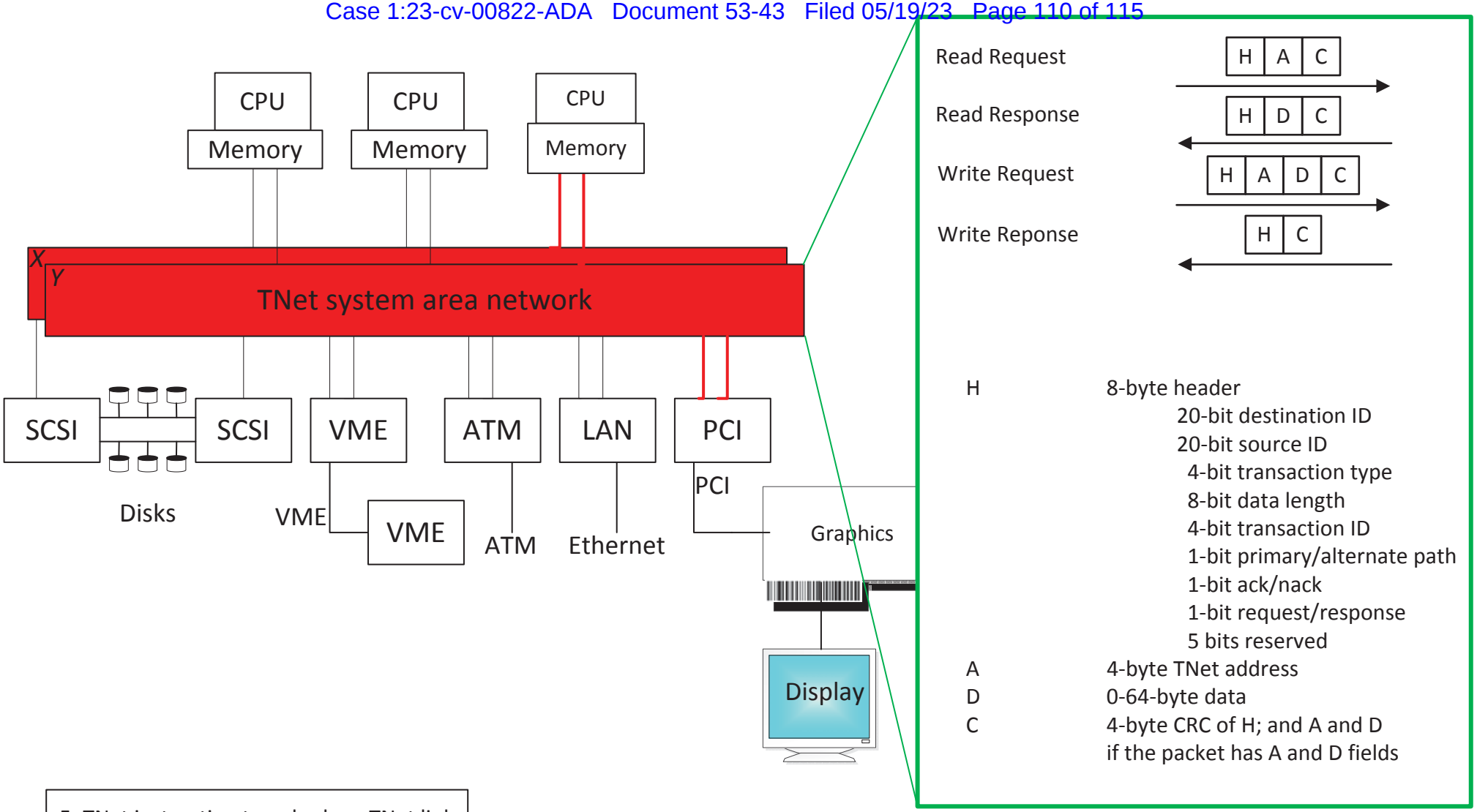
TNet Instruction Flow
Step 3



4: TNet Interface translates/converts Display Command to TNet instruction format

the TNet interface receives the Display Command in CPU command format. A CPU command format instruction cannot transmit over the TNet link. The TNet interface translates/ converts the Display Command to a TNet read or write transaction for transmission over the TNet link.

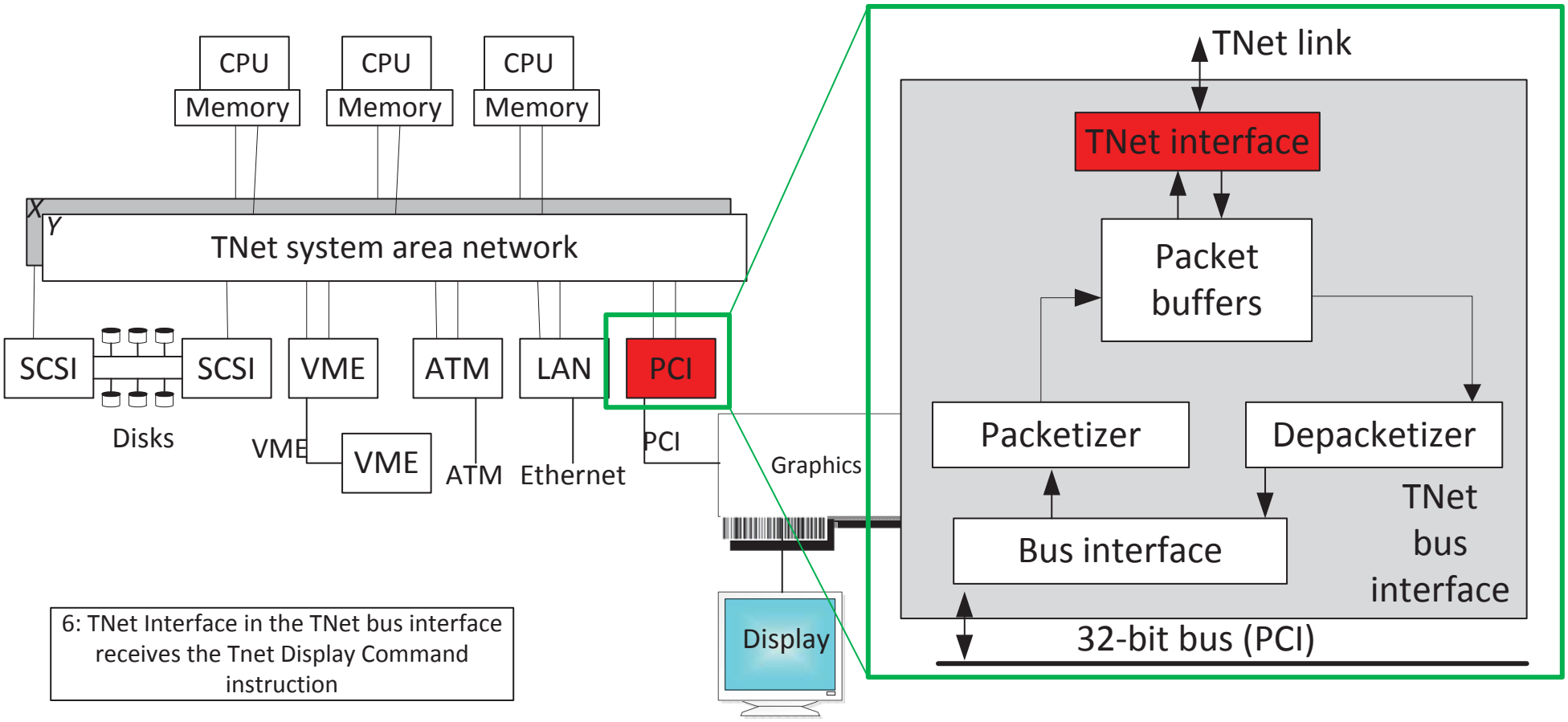
TNet Instruction Flow
Step 4



5: TNet instruction travels along TNet link to TNet bus interface for PCI

The Display Command is transmitted over the TNet link to the TNet bus interface for PCI. At this point, the Display Command is a TNet read or write transaction

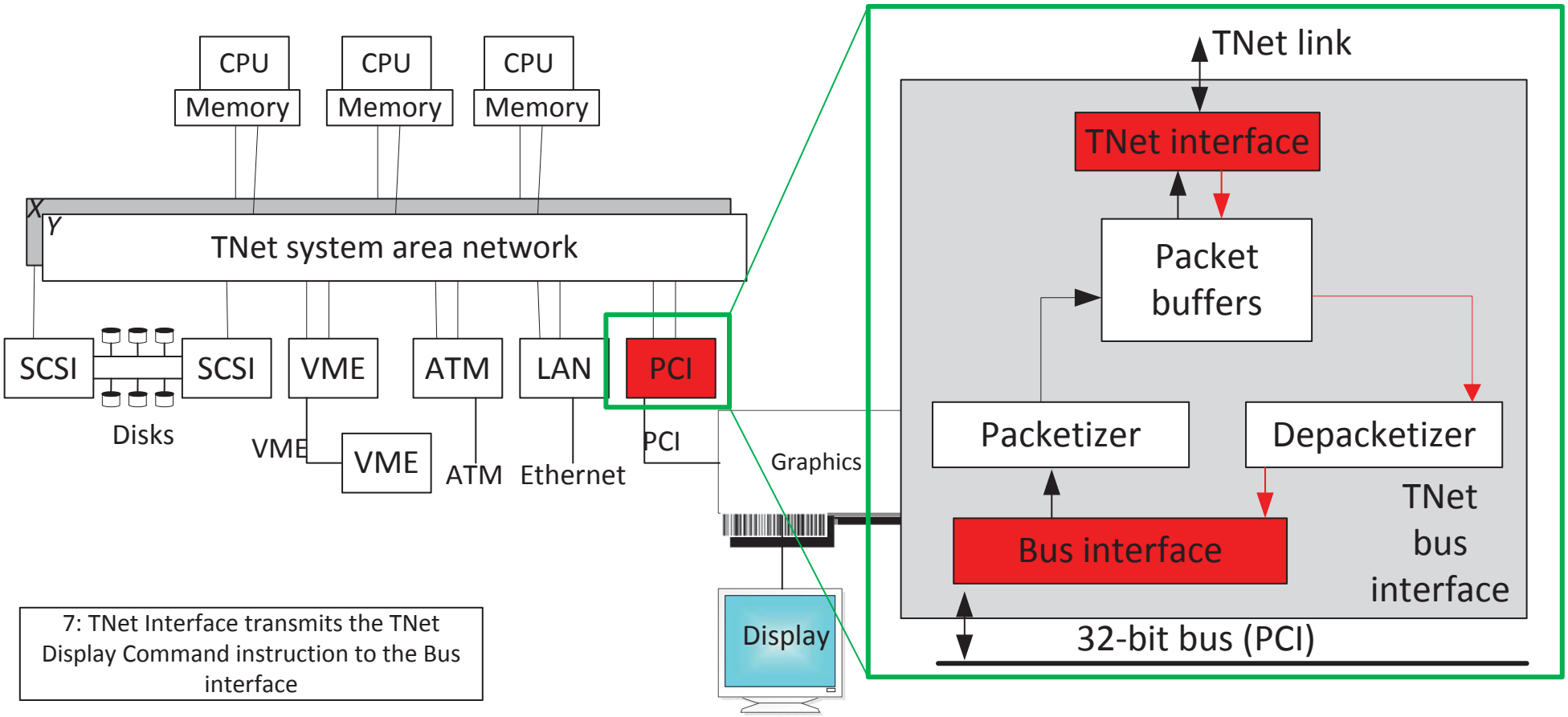
TNet Instruction Flow
Step 5



6: TNet Interface in the TNet bus interface receives the Tnet Display Command instruction

The TNet interface in the TNet bus interface receives the Display Command as a TNet read or write transaction.

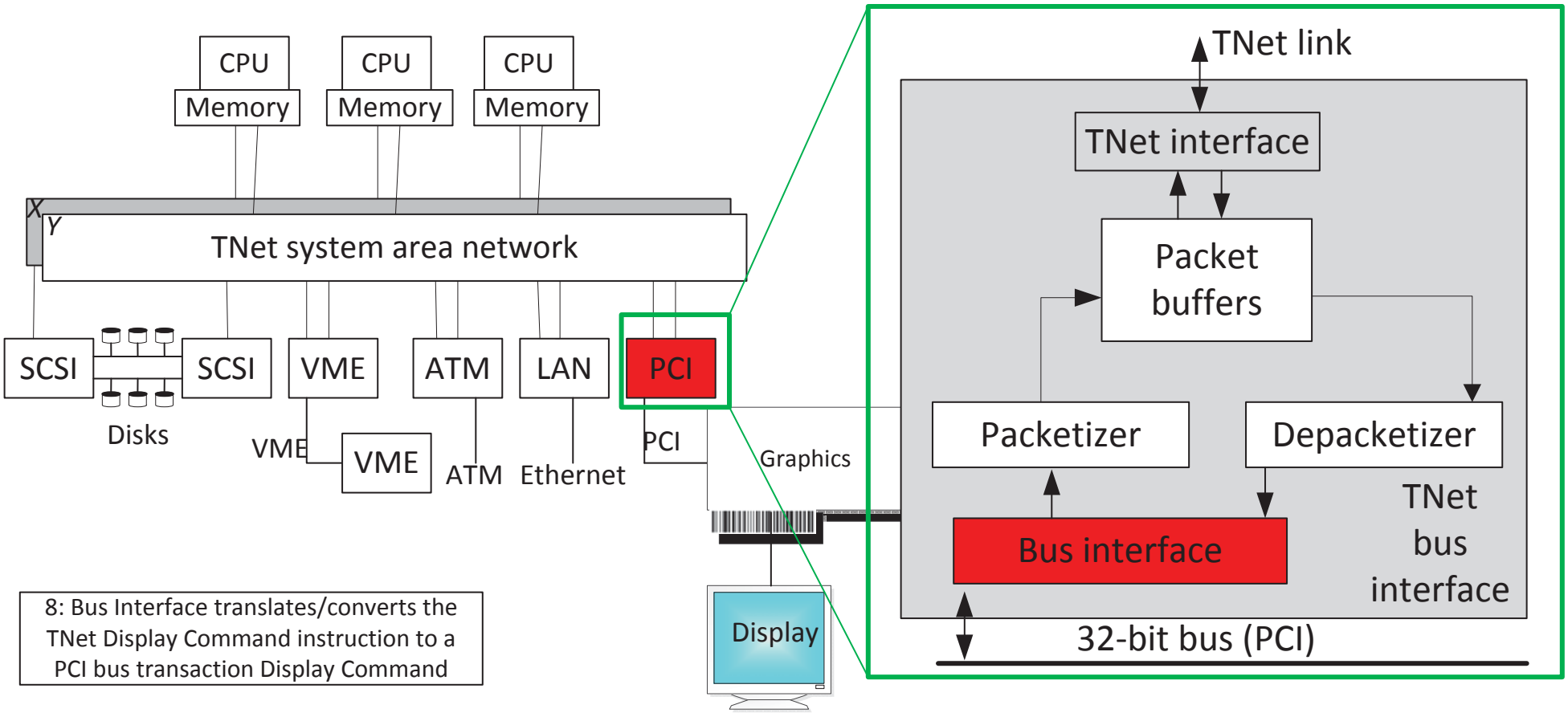
TNet Instruction Flow
Step 6



7: TNet Interface transmits the TNet Display Command instruction to the Bus interface

Upon receipt of the Display Command, the TNet interface sends the Display Command to the Bus interface within the TNet bus interface. At this point the Display Command is still a TNet read or write transaction.

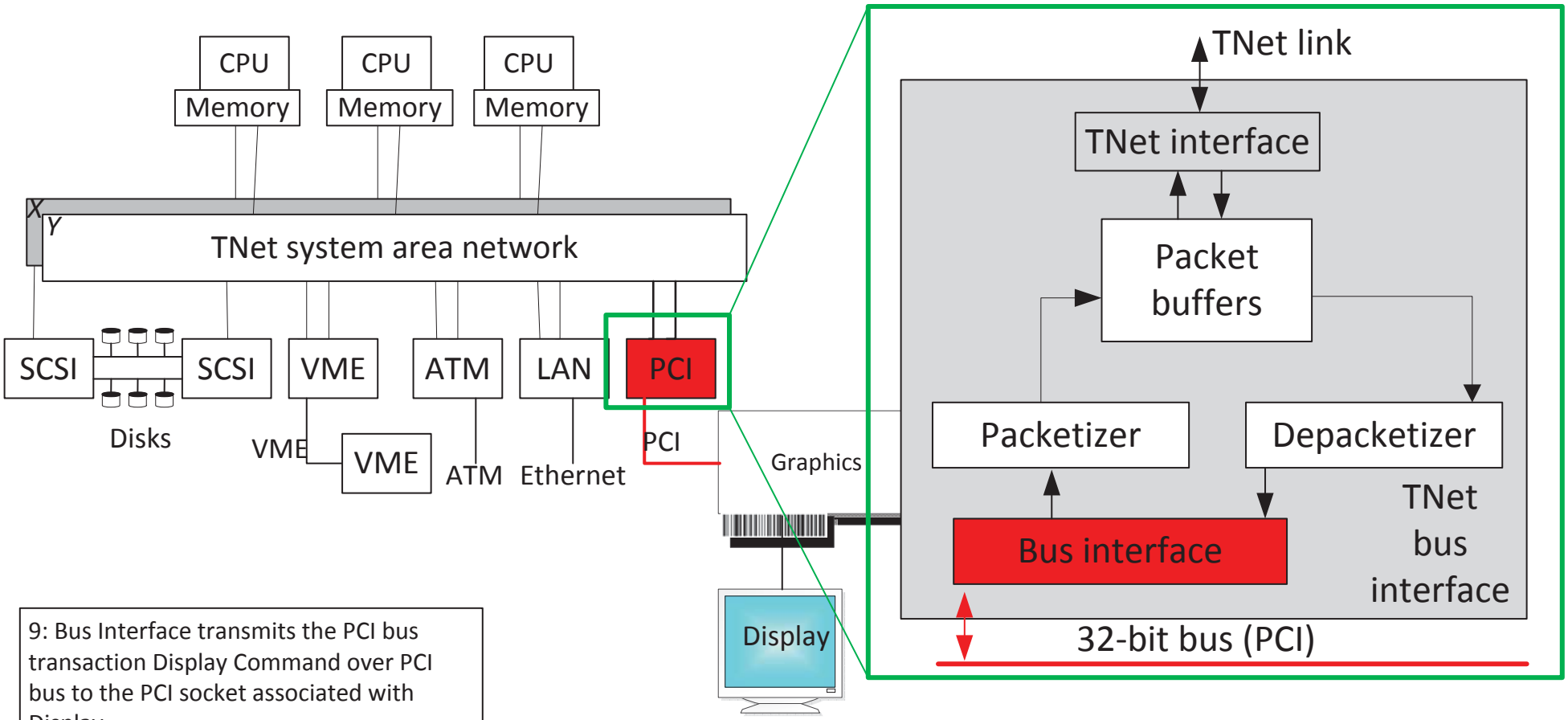
TNet Instruction Flow
Step 7



8: Bus Interface translates/converts the TNet Display Command instruction to a PCI bus transaction Display Command

The Bus interface within the TNet bus interface receives the Display Command as a TNet read or write instruction. A TNet instruction cannot be sent over the PCI bus. The Bus interface translates/converts the Display Command from a TNet instruction to a PCI bus transaction. This is the first time the Display Command is a PCI bus transaction.

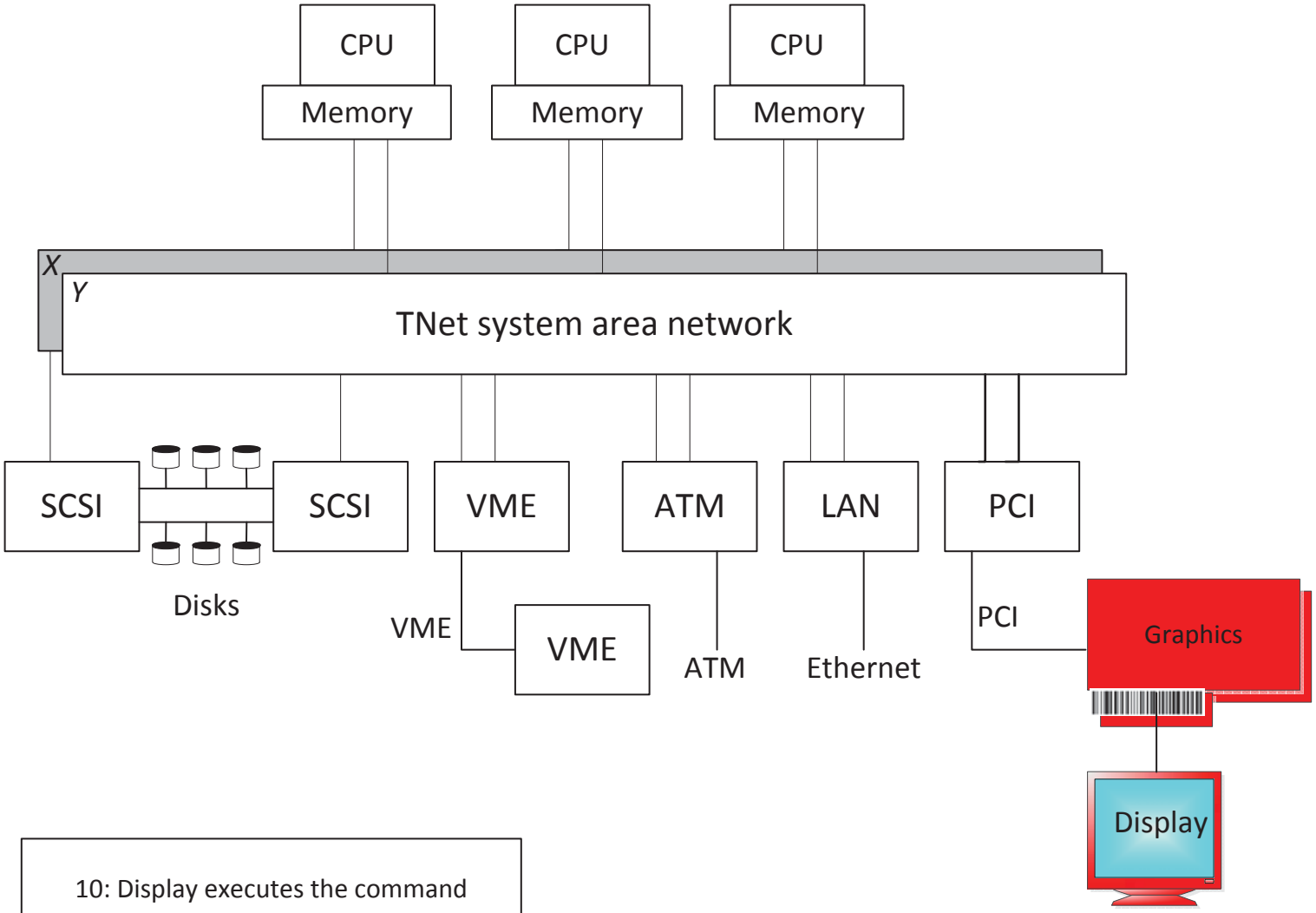
TNet Instruction Flow
Step 8



9: Bus Interface transmits the PCI bus transaction Display Command over PCI bus to the PCI socket associated with Display

The display command is now a PCI bus transaction, is transmitted over the PCI bus to the PCI socket associated with Display.

TNet Instruction Flow
Step 9



10: Display executes the command

Display receives the command and displays the information.

TNet Instruction Flow
Step 10